(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 712 097 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
15.05.1996 Bulletin 1996/20

(51) Int. Cl.[6]: **G06T 15/70**

(21) Application number: **95117529.8**

(22) Date of filing: **07.11.1995**

(84) Designated Contracting States:
DE FR GB

(30) Priority: **10.11.1994 US 337566**

(71) Applicant: **MATSUSHITA ELECTRIC INDUSTRIAL CO., LTD.**
Kadoma-shi, Osaka 571 (JP)

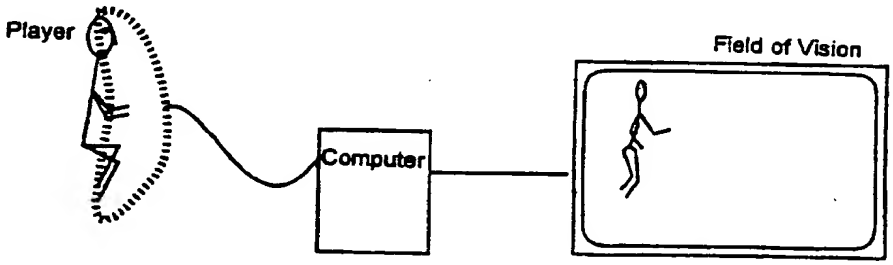
(72) Inventors:
- **Dow, Douglas Eric**
  2368 Singapore (SG)
- **Inada, Kazuhiko**
  Kadoma-shi, Osaka 571 (JP)

(74) Representative: **Grünecker, Kinkeldey,**
Stockmair & Schwanhäusser
Anwaltssozietät
Maximilianstrasse 58
80538 München (DE)

(54) Method and system for manipulating motion units for computer articulated figure animation

(57) A method and system for manipulating and mixing previously stored motion units for computer articulated-figure animation. The motion units are combined in a weighted average. The weights give a priority to certain joint rotations of a motion unit which constitute the essence of that motion unit. The motion units can be mixed in an asynchronous fashion, with no limit on the number of motion units that can be mixed together. Phrasing control assigns a weight to each frame in a motion unit, and helps to ease the transitions at the beginning and end of a motion unit. Frame-to-frame smoothing is also used. Fatigue of an articulated-figure can be simulated by the addition of a "tired" motion unit.

FIG. 6(a)



EP 0 712 097 A2

Description

FIELD OF THE INVENTION

The present invention relates to the field of computer graphics, specifically to articulated figure (AF) animation.

This invention is a method and a system to manipulate and mix previously stored Motion Units for computer AF animation. In the course of an application, such as a game or an educational title, the user directs the movement of the AF by selecting previously generated and stored Motion Units. Then, by the methods and systems of this invention, those Motion Units are combined in a weighted average with all the Motion Units that the user has selected to be currently running.

BACKGROUND OF THE INVENTION

A new generation of personal computers (PCs) and video game machines is now coming onto the market. These are capable of doing 3D graphics such as articulated figure (AF) animation. However, traditional research and approaches to computer AF animation has focused on the Animator. That is, they are focused on helping a professional Animator make some AF animation for a video or film that he/she is making. With this new generation of PCs and game machines, the user can now direct the AF animation; however, the above mentioned Animator-based algorithms are too computationally expensive. A professional Animator making a film can work off-line. The Animation does not have to be generated in real-time, and it does not have to be interactive. The Animator can develop each frame of the animation, taking just as much time as is necessary, then store the frame onto a storage medium. Later, all the stored frames can be played back rapidly at video speeds. However, interactive applications, such as games or educational software, must generate the 3D user-directed AF animation at, or close to, real-time. The methods and systems constituting the invention are aimed at the interactive game or educational applications which could use computer AF animation.

The focus of the invention centers on manipulating Motion Units. These Motion Units (MUs) define the articulated figure's orientation (joint rotations) for each frame in an animation sequence. The game or application creator uses separate methods (including rotoscopy, key frames, dynamics, and kinematics) to create an MU. Then the algorithms of the invention manipulate and run the stored MUs as the game user directs. The algorithms are simple and fast so as to not overburden these lower end machines (which are often required to generate one frame of video 30 times per second). Yet the motions that are created by manipulating and mixing the MUs are fairly smooth and natural looking. Above all, the user is in control.

The invention provides simple user-based algorithms which can rapidly manipulate previously scored MUs.

SUMMARY OF THE INVENTION

Firstly, concepts of the fundamental words of the present invention are explained. In the embodiments respective joints of the articulated figure have a parent-children relationship and can be expressed with a wooden structure. The each joint has local coordinate systems, and the rotation angle of the XYZ axes in the local coodinate systems of the child joint becomes the joint rotation, when the local coordinate systems of the parent joint is made to be a reference.

Motion Unit is an motion data of an articulated figure in an certain motion. When all the joint rotations of respective joints are determiend, and further when the length between two neighbering joints is determiend, the shape of the articulated figure is determined. By the joint rotation angle and the length the position of the joint are decided. The motion data has, for example in case of walking motion, retains the joint rotations of respective joints for one cycle.

The motion transition means, for example, the changing of the walking motion to punching motion.

The asynchronously selection means such selection that in the midway of, for example, the walking motion, if punching motion is selected, the articulated figure performs the motion walking and punching motion.

Previously stored Motion Units (MUs) that describe the joint rotations of an articulated figure (AF) for each frame in an animation sequence can be mixed together by a weighted sum calculation. The weights give a priority to certain joint rotations for a certain MU which constitute the "essence" of that MU. In other words, the joints of an MU that have been given a higher weight are deemed to be the more important joint rotations for this particular MU. The Animator who created the MU determines these joint weights. The goal is to determine the important joint rotations of an MU motion, and to give those joints a higher joint weight than the other joints. For example, in a punch motion for a human AF, the shoulder and elbow joints should be given a higher joint weight to stress the essence of the punch motion. Likewise, a kick MU should have the hip, knee, and ankle joints be assigned a much higher joint weight than unimportant joint rotations such as the wrist or the neck.

The MUs can be mixed in an asynchronous fashion, with no limit on the number of MUs that can be mixed together.

Phrasing control assigns a weight to each frame in a stored MU, and helps to ease the transitions at the beginning and end of an MU. This is accomplished by giving the frames at the beginning and at the end of an MU lower weights, and assigning higher weights to the middle frames.

Tiredness or fatigue of an AF can be simulated by adding a "Tired" MU with a weight proportional to the tiredness-factor to the current queue of concurrently running MUs.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 shows the overall application environment in which the invention applies.

Fig. 2 further describes the "player" block in Fig. 1. The player, or motion controller, can consist of a human or some type of artificial intelligence (AI) program that simulates a human player. Also a combination of the two is possible, where the human player is assisted by the AI program to control the movement of the articulated figure (AF).

Figs. 3(a)-(c)further describe the "interface" block in Fig. 1. Various interfaces are possible for the user to input and possibly receive non-vision sensory feedback from the computer. Examples shown are a hands-oriented device in Fig. 3(a). This could be a keyboard, button pad, joystick, data glove, or other such hand input device. Fig. 3(b) shows a legs-oriented input system where the position and movement of the feet, knees, and/or waist provides the input to the system. Fig. 3(c) shows an example of a possible feet, hands, and head combination interactive control system.

Figs. 4(a)-(b) and Figs. 5(a)-(b)further describe the "field of vision" block in Fig. 1. Fig. 4(a) shows an embodiment where the computer-generated scene is projected in front of the player. Examples of this include television, monitors, and projection systems. Fig. 4(b) shows an example of the screen shape that the player would see. Fig. 5(a) shows the possible configuration of a head mounted system. The computer-generated scene is projected close to the eyes, or even drawn directly onto the eye. Examples include flat screens and other head mounted vision systems. Fig. 5(b) shows two views of the scene that could be generated and displayed, one for each eye of the player.

Figs. 6(a)-(b) show one embodiment of the invention. In Fig. 6(a), one human player controls the motion of one AF which is in the field of vision. The motion control method is according to the invention. Fig. 6(b) shows a human player being assisted by an AI program. For example, the human inputs high level goals and motion desires, and the AI program converts those high level goals into the low level Motion Unit (MU) selections that are then merged and mixed according to the invention.

Figs. 7(a)-(b) show a further embodiment of the invention, involving multiple articulated figures in the field of vision. In Fig. 7(a), one human player controls the motion of multiple AFs. One example would be the player directing a sports team to accomplish some goal. Fig. 7(b) shows the combination of a human player and an AI program in controlling multiple AFs. In one embodiment of the invention, the AI program would assist the human player in directing and controlling the movement of the multiple AFs. In another embodiment, the human player would direct the movement of a partial group of all the AFs, while the AI program (or AI programs) directs the movement of other AFs.

Figs. 8(a)-(c) show more combinations of the embodiments of the invention. Fig. 8(a) shows multiple human players controlling multiple AFs. Either each player directs a different AF (or group of AFs), or multiple players work in unison as a team to control the motion of an AF (or group of AFs). Fig. 8(b) shows an example of all the inputs being AI programs and not human players. An example of this might be a simulation of a group of people such as a military scenario, emergency scenario, human traffic flow at an airport simulation, or other such study in groups of AFs, where the AFs behavior is determined by the corresponding controlling AI programs. Fig. 8 (c) shows an embodiment of the invention which uses a combination of human player (or players) and AI program (or AI programs) to control multiple AFs.

Figs. 9(a)-(c) further describe the "computer" block in Fig. 1. From the player's input through the interface, motion is generated for the AF(s). Then the Geometric Engine of Fig. 9(b) creates the 3D shape, orientation, and surfaces of the AF(s). Then the Image Generator shown in Fig. 9(c)converts those said 3D descriptions from the Geometric Engine into a 2D image for display in the field of vision of Fig. 1.

Figs. 10 (a)-(d) show various embodiments of a joint in an articulated figure for the invention. An articulated figure is a 3D shape described by the rotation at the joints. Fig. 10(a) shows an example of two static segments which pivot on a common axis which is the joint. Fig. 10(b) shows a ball-and-socket type joint. In this joint the segments can be oriented in any 3D rotation around the joint. Fig. 10(c) shows an AF shape that could be in a computer-generated world, where the segments are separated and do not physically or graphically touch, yet they rotate at a joint. Fig. 10(d) shows an example where the segments are not distinctly and separately defined; instead, the segments have merged into one, and the surface bends and rotates at the joint. An example of Fig. 10(d) is the human elbow joint, upper arm, and lower arm.

Figs. 11(a)-(b) show two examples of the skeleton of joints for the AF which are manipulated by the algorithms of the invention. The drawn figures in Fig. 11(a) and in Fig. 11(b) have a 3D surface which is not visible in the drawing. Fig. 11 only shows the joints (round black dot) and the relation between the joints as represented by the line connecting the two joints. The line can be considered to be similar to a skeleton of a human, or a frame, or segment-link of a robot.

Figs. 12(a)-(b) show two possible AF skeleton shapes that are different from any human-like shape. In the AF described in the invention, a composite 3D shape which can be described by 3D rotations at joints connecting the limbs or segments is an AF that can be manipulated by the algorithms described in the invention.

Fig. 13 shows one example of an outer surface of an AF. The joints and skeleton of the AF in Fig. 13 are similar to the ones shown in Fig. 11. The AF in Fig. 13 is moving according to the motion algorithms of the invention.

Figs. 14(a)-(c) further show the details of the invention as seen in Fig. 1. The "player" and "interface" are the same as that of Fig. 1. A "player" interacts through the "interface" in Figure 14(a). The "computer" receives the user's input which selects MU(s) to mix. These selected MU(s) are mixed in the "MU Mix Algorithm". Then the resultant frame is rendered and displayed in the "Field of Vision" that can be seen by the player. Fig. 14(b) shows the contents of the "MU Mix Algorithm" of Fig. 14(a). From the "MU Library", MU(s) are selected by the player and added to the "Queue of MUs" that are concurrently running. The MU(s) on the Queue are mixed in the "Mix" by a weighted average calculation. Then the resultant motion is smoothed. The "computer" block of Fig. 1 is subdivided into the rest of this diagram, Fig. 14(c). The overall application scenario is shown for the example of one player and one AF. The player selects MUs from a database (library) containing multiple MUs. An MU, asynchronously selected by the player, enters the queue of concurrently running MUs. By the algorithms of the invention, those MUs are mixed and merged to produce a description of each joint rotation of the AF. This is repeated frame after frame in an animation. The mixed motion can be further smoothed by a frame-to-frame smoothing technique described in the invention. The resulting motion description is entered into the Geometric Engine of Fig. 9, and is later drawn in the field of vision.

Fig. 15 shows possible ways the MUs in the MU Library of Fig. 14 are produced. The methods shown in Fig. 15 are outside the scope of the invention. These are accomplished previous to the application described by the invention. However, after the MUs have been generated by the methods of Fig. 15 (or other methods) and stored in the MU Library of Fig. 14, then the algorithms of the invention manipulate those stored MUs to produce AF motion as directed by the player.

Figs. 16(a)-(c) show three MUs being mixed according to the method of the invention to produce the resultant mixed MU motion for the AF. In this example, the bend-elbow MU that has previously been stored in the MU Library of Fig. 14 has been selected by the player to run concurrently with the wag-head MU and the bend-knee MU (also from the said MU Library). These motions are mixed by the weighted average calculation of the invention to produce the mixed MU shown at the bottom of Fig. 16(a).

Fig. 17 shows the file specification of an MU. An MU consists of a series of data for frames, one frame after another to describe the animation of the AF. Each of said frames consists of a long vector of rotation angles for each joint. In one embodiment of the invention, the stored rotation data for each joint consists of 3 angle values: (θ_z, θ_x, θ_y). This data can form a sort of matrix as pictured in Fig. 17, where there is a series of frames (F), and each frame consists of a series of joint data (J).

Fig. 18 shows an embodiment of the queue of concurrently running MUs as shown in Fig. 14(c). In the example seen in Fig. 18, there are two MUs currently on the queue of running motion units. The actual data on the queue stored in this embodiment of the invention consists of a data structure that contains at least two items: a point to the actual MU data in the MU library, and the current frame number (F). This frame number keeps track of the current frame in the MU that is to be run next. The weighted joint angle calculation of the invention refers to this frame number (F) to determine which actual frame of the MU needs to be mixed with other frames from the other MUs on the queue of running MUs.

Fig. 19 shows various embodiments of the ramp weights that are in the invention. Basically, in the preferred embodiment, the initial frames of an MU have a lower weight, the middle frames of the MU have a relatively higher weight, and the final frames have a lower weight. The effect of these frame weight patterns is to enhance the smoothness of the transitions at the beginning and end of MUs so that the resulting mixed motion is smoother. Also these said frame weights can be used to manipulate and stress important frames of the MU.

Fig. 20 shows another embodiment of the said frame weights (some of the other embodiments are shown in Fig. 19). In Fig. 20, N_m is the number of frames in the MU to which this frame weight pattern is applied. N_r is the number of frames that linearly increment up from the low weight to the higher weight. The converse is true for the final N_r number of frames in this said MU.

Fig. 21 shows an example of the invention mixing MUs. A player has asynchronously selected the three MUs in Fig. 21: Kick MU first, then later the Punch MU, and later the Nod-Head MU. Each of these said MUs has a frame weight pattern as seen in Fig. 20. For example, the said Kick MU's first frame is executed at time t_a, while the last frame at time t_b. In addition to the said three MUs that the player asynchronously selected, a continuously running Idle MU (jog MU) is always running. This Idle MU has a comparatively low frame rate for all its frames, and it serves in this embodiment of the invention as a background motion for the AF to move according to when no other player-selected MUs are running. For example at time t_o, the player has not selected any MU to run yet (the Kick MU starts later at time t_a), so the said Idle MU is the only MU running at time t_o.

Fig. 22 shows a simulation of the fatigue that is possible in one embodiment of the invention. A background MU similar to the above said Idle MU of Fig. 21 is added to the queue of running MUs. As a value called tired (fatigue) rises from 0 to 1, the weight of this looping tired MU increases from 0 to 1 causing the AF to behave more and more according to the tired MU motion.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

The preferred embodiment of the invention is in an interactive computer graphics game involving one or more animated articular figures, or articulated figures, that are visible to the player and are controlled by the player. The player asynchronously selects MU(s) for the articulated figure to do. Then the algorithms specified by the invention mix the concurrently running MUs selected by the player. The result of this mix calculation is the rotation of each joint for the next frame in the animation. The articulated figure is positioned in that position (as specified by the calculated rotation angles) and rendered so that they are visible to the players. In the preferred embodiment, the articulated figures are made visible by displaying them on a video monitor(s) that is (are) visible to the player.

Another embodiment of the invention can be a simulation of one or more articulated figures. The purpose would be to make an animation of articulated figures moving or to simulate some behavior of articulated figures as determined by some behavior or artificial intelligence (AI) algorithm that selects MU(s) for the articulated figures.

Another embodiment of the invention is the controlling of the motion of articulated figures in a Virtual Reality (VR) application. Either humans or software modules (AI Program Agent in Fig. 2) select the next MU for the articulated "figure to start doing.

The invention allows the user to directly control the mixing of previously stored Motion Units (MUs). Examples of a Mu could be *Bow. Wave. Shake Head. Walk. Punch.* or *Kick.*

Before the game application starts, the MUs must have been created and stored. The boxes in Fig. 15 indicate some of the different techniques that have been developed to make realistic looking human motion. These methods have been developed and used by many researchers, and some are even commercially available. The motion generating methods include rotoscopy (described in Kunii T.L., Sun L., "Dynamic Analysis-Based Human Animation", Proc. Computer Graphics International '90 (CGI '90), Singapore, 1990, pp. 3-15; Unuma M., Takeuchi R., "Generation of Human Motion with Emotion", Proc. Computer Animation '91, Geneva, Switzerland, Springer-Verlag, 1991, pp. 77-88 and which are hereby incorporated by reference for their teachings on rotoscopy), key-frame (described in Steketee S.N., Badler N.I., "Parametric Keyframe Interpolation Incorporating Kinetic Adjustment and Phrasing Control", Proc. SIGGRAPH 1985, Computer Graphics, Vol. 19, No. 3, pp. 255-262; Burtnyk N., Wein M., "Interactive Skeleton Techniques for Enhancing Motion Dynamics in Key Frame Animation", Graphics and Image Processing, Association for Computing Machinery, 1976, pp. 564-569 and which are hereby incorporated by reference for their teachings on key-frame), dynamic procedural (described in Noser H., Thalmann D., "L-System-Based Behavioral Animation", Proc. Pacific Graphics '93, Seoul, Korea, Computer Graphics and Applications, World Scientific, Vol. 1, pp. 133-146 and which is hereby incorporated by reference for its teaching on dynamic procedural), kinematics (described in Korein J.U., Badler N.I., "Techniques for Generating the Goal-Directed Motion of Articulated Structures", IEEE Computer Graphics & Applications, Nov. 1982, pp. 71-81 and which is hereby incorporated by reference for its teaching on kinematics), AI procedural (described in Badler N.I., Webber B.L., Kalita J., Esakov J., "Animation from Instructions", Making Them Move: Mechanics, Control, and Animation of Articulated Figures, ed. Badler, Barsky, Zeltzer, Morgan Kaufmann Publishers, 1991, pp 51-93; Morawetz C.L., Calvert T.W., "Goal-Directed Human Animation of Multiple Movements", Proc. Graphics Interface '90, Canada, 1990, pp. 60-67. and which are hereby incorporated by reference for their teachings on AI procedural), and various interactive mixed mode methods (described in Boulic R., Thalmann D., "Combined Direct and Inverse Kinematic Control for Articulated Figure Motion Editing", Computer Graphics Forum, Vol. 11, No. 4, 1992, pp. 189-202; Calvert T., "Composition of Realistic Animation Sequences for Multiple Human Figures", Making Them Move: Mechanics, Control, and Animation of Articulated Figures, ed. Badler, Barsky, Zeltzer, Morgan Kaufmann Publishers, 1991, pp. 35-50; Armstrong W.W., Green M., Lake R., "Near-Real-Time Control of Human Figure Models", Proc. Graphics Interface '86, Vision Interface '86, Canada, 1986, pp. 147-151; Calvert T., Bruderlin A., Dill J., Schiphorst T., Welman C., "Desktop Animation of Multiple Human Figures", IEEE Computer Graphics & Applications, May, 1993, pp. 18-26; Boulic R., Huang Z., Magnenat Thalmann N., Thalmann D., "A Unified Framework for the Motion Manipulation of Articulated Figures with the TRACK System", Proc. CAD/Graphics '93 Beijing, New Advances in Computer Aided Design & Computer Graphics, Ed. Tang Z., International Academic Publishers, Vol. 1, 1993, pp. 45-50 and which are hereby incorporated by reference for their teachings on various interactive mixed mode methods). Much energy, development, and refinement has gone into each of these methods of generating motions.

Moreover, any other method that has been developed in computer human animation research is an appropriate method to generate an MU if the following condition is met. The motion specified by the created MU must be of the form, or be able to be converted to the form, of the data structure shown in Fig. 17. That is, the motion is specified frame by frame. Each frame is specified by a series of angles that represent the rotation of each of the rotatable joints of the articulated figure. In the techniques shown in Fig. 15, the specification of an angle joint consists of three angle rotations. However, any scheme of joint specification is covered by the methodology of the invention so long as a particular joint of the articulated figure is specified by an angle, or a series of angles, in a consistent way from frame to frame. The angle's values must be of a type that can be manipulated by the mathematical equations shown in equation (1) (below) of the invention.

After generating a desired motion by one of the above techniques, the application developer needs to store that motion frame by frame. The key data to be stored is each joint's rotation information.

There are a number of ways to specify the orientation of a joint. The motion mixing algorithms specified in the invention work best if the data consists of the rotation angle of the joint, where this angle is relative to itself (i.e. flexing, pivot, twisting), or relative to its parent joint's local coordinate system. Fig. 17 shows that an MU consists of rows of frames where each frame consists of a sequence of joint rotation data.

As shown in Fig. 1 the invention relates to the environment of a "player" who is interacting through an "interface" with a "computer". Frames of visual images are fed back from the computer to the player. The invention is an algorithm being executed in the computer box of Fig. 1.

As shown in Fig. 2, in the preferred embodiment the player (of Fig. 1) is a human, but the player specified in the invention could also be a software or other electrical/mechanical entity that is simulating a human player's input to the interface and to the computer of Fig 1. For example, instead of a human player selecting the next MU for the articulated figure to execute, a software module could select, by its internal program, the next MU for the articulated figure to execute. In this way, the "player" described in the invention can be a software program. This software module that is the "player" is labeled "AI Program Agent" in Fig. 2, Fig. 6(b), Fig. 7(b), Fig. 8(b), and in Fig. 8(c).

As shown in Fig. 3, the interface (of Fig. 1) is a device that is manipulated by the hands. This would include all the currently used paradigms of interaction such as keyboard, joystick, data glove, button pad, and any other device that can be controlled by movements of the hand or hands. In addition, the interface (of Fig. 1) of the invention can be a device that is manipulated by the motion of body parts in other regions of the body such as the legs and/or feet, and motions of the face, eyes, head, ears, nose, neck or shoulders. If the player (of Fig. 1 and Fig. 2) is a software module, the interface is a device capable of detecting the commands of the software module. The commands could be software procedure calls, method calls, computer messages, electronic mail, light signals, and other electrical and/or mechanical medium for the AI Program Agent to input its commands through the interface (of Fig. 1) to the computer (of Fig. 1). The important thing is that the interface of this device is receiving input from the player (of Fig. 1) and converting that input into AI selections for the articulated figure to execute.

As shown in Fig. 4, the Field of Vision (of Fig. 1) displayed to the player (of Fig. 1), specifically a human player, is a video monitor in the preferred embodiment of the invention. In addition, other visual systems that are further embodiments of the invention include head mounted visual systems such as those used in Virtual Reality applications, as shown in Fig. 5.

An illustration of the preferred embodiment of the overall system can be seen in Fig. 14(c). In this figure, the user asynchronously selects various MUs to run. These MUs are added to the queue of concurrently running MUs. Each animation frame, one frame of joint data from each MU, is mixed in the "Mixer" seen in Fig. 14(c). Here a weighted average of the joint's frame is calculated to determine the resultant joint rotation values for that frame. Each joint data element in an MU contains a weight which indicates its priority. Specific AF motions (i.e. jump, sit)are accomplished through the movement of key joints. Of course, many other non-key joints may also move, but they are not essential to the movement. For example, in a jump, the leg joint angles are essential, the arms are somewhat important but the neck or wrist joint angles are not important at all to the essence of a jump.

The preferred embodiment of the invention pairs one player to one articulated figure as shown in Fig. 6(a). The player selects MUs for the articulated figure to execute, and the articulated figure uses the resultant motion that was determined by the algorithms of the invention based on the selected MUs of the player. Moreover, a human player may work in cooperation with a computer module, the AI Program Agent, as seen in Fig. 6(b). For example, the human would select high level motions that are timed combinations of multiple MUs. The AI Program Agent provides the specific selection of MU at the appropriate time in order to fulfill the expressed motions desired by the human player. Another scenario is for the human player to select MUs, and also the AI Program Agent to select MUs. Then these two sets of MUs are mixed by the algorithms of the invention to produce the resultant motion.

More embodiments of the invention are shown in Fig. 7. In Fig. 7(a), one human player controls a group of articulated figures. One example of this group and coordination could be a team such as a sports team. The player selects MUs which instruct each articulated figure of the game how to move. In Fig. 7(b) the human player works in cooperation with an AI Program Agent to coordinate the movement of the multiple actions. For example, the human player could specify high level motions while the AI Program Agent specifies the lower level selection and timing of MUs to execute to stated high level motions as specified by the human player. Another scenario would have the human player controlling one or one group of the articulated figures, while the AI Program Agent controls other articulated figures. The two sets of articulated figures could be working in cooperation or competition.

Fig. 8 shows more combinations of embodiments of the invention. Fig. 8(a) shows an example of multiple human players controlling a multiple set of articulated figures. The relation between a player and an articulated figure could either be a one to one, or a one to many relation. That is, the MUs selected by a human player could control just one articulated figure, or they could control a group of the articulated figures. Fig. 8(b) shows a further embodiment. This depicts a scenario where no human player is involved, only computer players, the AI Program Agents. An example of this is a simulation of a group of articulated figures. The AI Program Agents could be programmed to select MUs for the

articulated figures that make them move in a way that simulates a situation such as a war game, crowd behavior, or human ergonomics study. This embodiment could be used for a research purpose in order to study and learn about the simulated articulated figure behavior. Another simulation example would be an animation simulating people moving in a design proposal of a new building, store, park, or factory. The articulated figures whose motions have been determined by the MU selected by the AI Program Agents would move in a computer graphics simulation of the design proposal of the new building. A video of this animation could then be shown to potential buyers, bankers, customers, politicians, or other interested people. Such simulations of articulated figure behavior are an embodiment of the algorithms of the invention. Likewise, Fig. 8(c) shows an embodiment of the invention which has a combination of multiple human players and multiple AI Program Agents selecting the MUs and thereby controlling the motions of multiple articulated figures. The multiple players, in any combination, can be acting in cooperation or acting in competition.

Fig. 9(a) shows the computer (of Fig. 1) in the preferred embodiment. The algorithm of the invention would be in the box labeled "Generate motion for Articulated Figure". Note that Articulated Figure means the same as articulated figure (of Fig. 1) in the documentation of the invention. For the computer to then convert for each frame the calculated resultant motion of each articulated figure to the Field of Vision (of Fig. 1), a Geometric Engine (of Fig. 9(b)) and an Image Generator (of Fig. 9(c)) are in the preferred embodiment.

The Geometric Engine of Fig. 9(a) is expanded in Fig. 9(b). The geometric engine is described in Foley J.D. and Van Dam A., "Fundamentals of Interactive Computer Graphics", Addison-Wesley Publishing Co., 1984, pp 262-265, which is hereby incorporated by reference. Here, the body of the articulated figure (AF) is constructed in a model coordinate system and then transformed to the world coordinate system. The input to the Geometric Engine consists of all the 3D angle values for each joint relative to its neighbor joints in the AF. These 3D angle values are the output of the algorithms specified by this invention and represented in Fig. 9(a) as the block labeled "Generate Motion for Articulated Figure". From these angle values, the 3D position of each joint is calculated. A necessary ingredient to this calculation is the length of each segment between two joints. This length and the connectivity information (i.e., which joint is connected to which other joint) are contained in the block of information depicted as the computer disk labeled "skeleton" in Fig. 9(b).

The output data stream of the first function in Fig. 9(b) is all the 3D positions of each joint for one frame in the animation. These 3D joint positions are input to the second function in Fig. 9(b). This second function determines the surface of the tissue based on the neighboring joint positions. This is explained in Dow D., Semwal S., Maehara, F., "Modeling Complex Human Shape Using Flexible Object Oriented Methodology", Proceedings of NICOGRAPH '92, Tokyo, November 1992, pp 35-43, which is hereby incorporated by reference. The additional model data required to determine the surface of the tissue is obtained from the data block depicted in Fig. 9(b) as a computer disk labeled "tissue". Given the 3D position of the surrounding joints, the surface of that body region can be calculated and represented in various ways, including simple geometric shapes, polygon meshes, free form surfaces, or volume voxels. Each of these representations of the surface tissue is an appropriate way to use the output data produced by the algorithm of this invention.

Finally, this calculated surface representation data is passed from the second function of Fig. 9(b) to the third function. In this third function, the representation of the surface of each body region of the AF is correctly positioned and oriented in the "world coordinate system" of the computer scene. In simple terms, this transformation puts the AF in the correct place and orientation in the scene currently being displayed on the computer display. A further explanation of this transformation of coordinate systems that is used in 3D computer graphics systems can be found in Foley and Van Dam.

The output data stream of the Geometric Engine is the output of the said third function of Fig. 9(b). This can be considered to be a display list of 3D points representing the basic graphic primitives that will be drawn on the computer screen. Such primitives include vertex points, line segments, polygon surfaces, and volume voxel density values.

The Image Generator of Fig 9(a) is expanded in Fig. 9(c). The image generator is described in Foley and Van Dam, cited above, pp 267-274, which is hereby incorporated by reference. The input data to this Image Generator are the 3D world coordinate output primitives, such as vertexes, lines, and surfaces. The Image Generator converts these 3D coordinates into a 2D projection plane. Finally, the 2D projection plane coordinates are rasterized to fill in the pixel values in a 2D frame buffer. The frame buffer data is in a form that can be directly sent to the display for viewing by the human player.

The articulated figures in various embodiments of the invention could have various joint configurations and various body surface methods. Fig. 10 shows four different ways to model or represent a joint in the articulated figure. Each of those joints (in Fig. 10) is an acceptable embodiment of the invention. The important thing is that regions of the articulated figure have motions that can be modeled by rotations about the center of the joint. As shown in Fig. 11 and Fig. 12, the number of joints or the configuration of joints in the articulated figure can be of various embodiments. The invention uses mathematical weighted average calculations as described in this document to manipulate the angle rotations of corresponding joints of concurrently running MUs to determine the resultant angle of that joint. So the number of joints in an articulated figure, or the configuration of the joints to form the articulated figure, can vary between different embodiments of the invention, so long as the MUs and the articulated figures have the same number of joints and angle values.

To encode the essence for each motion, the joints are given a weight. This weight determines the joint's relative priority to the purpose of a particular movement. When two or more MUs are mixed together, a weighted average of the combined joints determines the resultant joint angles.

$$\theta_J = (\Sigma^n_{i=0} w_{iJ}\ \theta_{iFJ}) / (\Sigma^n_{i=0} w_{iJ}) \quad (1)$$

where:

θ_j = rotation of J-joint, the calculated resultant angle

i = which MU from queue of running MUs

J = which joint in AF

n = number of MUs currently on the queue

θ_{iFJ} = rotation values of the i-MU's F-frame and J-joint

w_{iJ} = joint weight: weight for the J-joint of the i-MU

The above J in the equation is the same as the J Joints seen in Fig. 17. Each frame F of an MU has rotation data for each joint of the AF. Likewise, the above F in the equation is also the F Frames seen in Fig. 17. The MU_index in Fig. 18 keeps track of which frame F is the next frame of the MU to mix in the above equation.

If all $w_{iJ} = 1$, then the final rotation value is simply the average of the rotation values for each joint from each frame to be mixed. The weights give a priority to the joint rotations. This way, when two or more motions are merged, the joint rotations that have a higher weight have a higher priority and they more strongly influence the resultant motion.

Consider an example which combines three simple motions. One MU is a kick (bend-knee), one is a punch (bend-elbow), and the other is a head shake (wag-head). Fig. 16(a) shows the original three motions and the composite result.

In the kick motion, the leg joint motions are essential to the "kick" motion, so they are given a higher weight of 1.0, the rest are given a low weight like 0.1. Likewise, with the punch, the arm joints are given a weight of 1.0, and the rest are given a weight of 0.1. With the head shake, the head's weight is 1.0, and the rest are given a weight of 0.1. When these motions are run at the same time, a weighted average is calculated to determine the resultant joint angle from the three input MUs.

Consider a leg joint. For some frame in the motion sequence, leg joint weight is:

Kick MU: w = 1.0

Punch MU: w = 0.1

Head Shake MU: w = 0.1

$$\theta_{Leg} = (w_{Kick}\ \theta_{Kick} + w_{Punch} + w_{HdShk}\ \theta_{HdShk}) / (w_{Kick} + w_{Punch} + w_{HdShk}) \quad (2)$$

$$= ((1)\theta_{Kick} + (0.1)\theta_{Punch} + (0.1)\theta_{HdShk}) / 1.2$$

The kick's weight of 1.0 is much bigger than the 0.1 weight of the other MUs. Therefore, of the three MUs to be mixed, the kick MU dominates leg joints.

As seen in Fig. 14, the motion mix algorithm described above computes a weighted average of joint angles to be merged. However, since the user selects new motions asynchronously, there could be a frame-to-frame discontinuity that would produce an unnatural motion. This discontinuity problem during the transition from one dominate motion to the next has been called Badler phrasing (described in Steketee S.N., Badler N.I., "Parametric Keyframe Interpolation Incorporating Kinetic Adjustment and Phrasing Control", Proc. SIGGRAPH 1985, Computer Graphics, Vol. 19, No. 3, pp. 255-262 and which is hereby incorporated for its teaching that a motion unit is like a phrase of words).

To solve this phrasing problem, the invention uses a weight factor that is a variable of time. The weight gradually increases from zero at the start of an MU and finally decreases at the end. This minimizes the rough transition between two MUs. Fig. 19 shows various possibilities for how the weight of an MU frame could change over time. To minimize rough transitions, the frames near the start and near the end of an MU are made small. The preferred embodiment of the phrasing weight can be seen in Fig. 20.

An MU is defined from frames 0 to N_m in Fig. 20. In the first few frames of the motion, the frame has a low priority. But gradually, frame by frame, the weight is increased. Likewise, at the end of an MU, the weight of the last few frames diminishes. Therefore, at the beginning and end of the MU, the frames have a low priority, so they do not have much

influence on the resultant composite motion. However, the middle frames are at their full priority, so they do significantly influence the composite.

The original Equation (1) does not include this frame weight. So the more complete equation is:

$$\theta_J = (\Sigma^n{}_{i=0}\, w_{iF} w_{iJ} \theta_{iFJ}) / (\Sigma^n{}_{i=0}\, w_{iF} w_{iJ}) \quad (3)$$

where:

w_{iF} weight for i-frame (weight applies to all joints in frame)

Basically, the weight of a frame is multiplied by the weight of a joint to produce the product resultant weight of that joint. In the example shown in Fig. 16(a), each resultant "mixed MU" time step (t_0, t_1, t_2, ...) is calculated independently. The inputs to this calculation for t_i are the t_i "bend-elbow MU", the t_i "wag-head MU", and the t_i "bend-knee MU" joint angle values. The angle of each joint is calculated independently of the other joints. This follows directly from Equation (3). This Equation (3) is expanded in Fig. 16(b) so that it directly corresponds to the example of Fig. 16(a). For the calculation determining the angle of a single joint for a single step t_i, the product from each of the three MUs is summed up.

The flow chart in Fig. 16(c) further explains Fig. 16(a), Figs. 14(a)-(c) and Equation (3). The largest loop in Fig. 16(c) executes once per time step. That is, one time for each frame in the animation of the AF in motion. The next smaller loop executes once for each joint in the AF. This performs the calculation of Equation (3).

Fig. 16(c) also shows the existence and functionality of the "Motion Unit Library" and the queue of "Concurrently Running MUs". As shown in Figs. 14(a)-(c), when the player selects a new MU to start, that MU is inserted from the Motion Unit Library into the queue of Concurrently Running MUs. When this new motion is completed (probably many time steps, t_i, later) it is taken off the queue of "Concurrently Running MUs". Therefore, this queue is dynamic. The MUs on the queue may change each time step. More MUs might be added, and complete MUs will be taken off the queue. The smallest, innermost loop of Fig. 16(c) executes once for each MU that is on the queue of "Concurrently Running MUs".

Before considering a further example of how the weights from many different MUs interact, it is important to understand the idle motion.

At the bottom of the priority list of running MUs is always the idle MU. For each application the idle would be a different type of motion. For instance, a sports game might have the idle MU be a jog-in-place motion. But, an animation explaining cooking or wood-working might have the idle MU describe a standing position with the hands held about 10 cm above the table top.

The idle MU has a very low priority weight, so if another MU is running, that MU will dominate. However, the idle MU dominates when no other MU is currently running. During the transition at the end of a MU, the MU's weight priority for the last few frames becomes less than the priority of the idle MU, so a transition occurs between the ending MU, and the idle MU.

To better explain the said phrasing control and the said idle MU of the invention, an example of the preferred embodiment will be discussed. Consider the example in Fig. 21. This shows the queue of MUs that the user has selected to run. The following list explains which MUs are mixed at what times and which MUs dominate.

t_0 No MUs are selected by the user, idle MU dominates

t_1 Kick MU dominates

t_2 Kick MU and Punch MU both have low weights. Mix of those and idle MU

t_3 Punch MU and Nod-Head MU equally dominate

Here motions start and stop asynchronously. Concurrently different MUs run, and the net result is based on a weighted average calculation of the joints in each MU that are being mixed.

One embodiment of the invention simulates an emotional state for an AF. A value determines the strength of this emotional state in how much it affects the resultant outcome of the AF motion. For example, the emotional state of being tired, exhausted, or lazy can be simulated. Another background MU is added to the queue of running MUs. This MU is always on the queue because the MU_index of Fig. 18 is set so that the frame number returns to the first frame in the MU after the last frame in the MU has been executed. This is the same as used by the said Idle MU: it loops on the queue, never ending.

As a real human becomes more and more tired, he tends to move less vigorously and more lethargically. That is, as one becomes tired, movement is minimized, thereby conserving energy.

To implement the tired simulation in one embodiment of the invention, basically, a weighted MU representing an exhausted body position is added to the queue of MUs being mixed. If the AF is healthy and not tired, the tired-MUs weight is 0, so it has no effect on the resulting motion. But as the AF becomes increasingly tired, the weight of this tired MU increases from 0.0 to 1.0.

For example in Fig. 22(a), a normal (tired = 0.0) MU is shown. In (b) the actor is more tired (tirede = 0.5) so the motion is less energetic. In (c) (tired = 1.0) the motion is very lethargic.

In addition to the above said phrasing control, an additional frame to frame smoothing technique is optionally a part of the invention. As seen in Fig. 14(c), for each frame in the animation, the motion mixer combines one or more frames

in a weighted average calculation to produce this resultant frame. Even though this is a weight average of the joint weights, and phrasing control weights have been used to minimize the discontinuity at the start and end of MUs, there could still be some frame-to-frame discontinuity. Spline smoothing of animation frames could be used to smooth this.

Each application has different requirements of smoothness of motion, and constraints on processing power. Many simpler applications will skip this frame spline calculation and directly animate the character as calculated by the weighted sum calculation. Others will do a more sophisticated spline smoothing.

A compromise of the demands of smoothness and low computational power can be achieved by using a fixed spline calculation. This reduces the somewhat complicated spline calculation to be just another weighted average calculation. The preferred embodiment of the invention uses a frame-to-frame smoothing filter based on a nonperiodic B-Spline using three frames' joint rotation values. The B-Spline derivation can be found in other works (described in Mortenson M.E., Geometric Modeling, John Wiley & Sons, 1985, pp. 125-147 and which is hereby incorporated by reference for its teachings on B-Spline derivation). The equation for a nonperiodic B-Spline using three points:

$$p_i(u) = 1/2[(1-u)^2 p_{i-1} + (-2u^2 + 2u + 1)\, p_i + u^2 p_{i+1}] \quad (4)$$

This calculates a smooth curve between the three points for a range of in from 0 to 1.0. The goal is to ensure continuity as MUs are being mixed and running concurrently. Various types of spline calculations could do this. The preferred embodiment uses the above described B-Spline that determines a point at u = 0.75 for each frame in the animation.

Another advantage of using splines in this way is that each calculated frame can be considered as a key-frame. Then the well developed algorithms of key-frames can be used to calculate any in-between frame that is desired. This allows the application to have control of the output frame rate, regardless of the frame rate of the stored MUs that is being used. In-between frames can be calculated to speed up or slow down the animation.

Nevertheless, the major goal of all the motion algorithms presented in the invention is to provide a fast way to allow user-based AF animation in low end platforms such as PCs or games. Therefore, in many cases, the discussion in this section about splines (to ensure continuity between frames) may be too slow a calculation for many platforms. If so, this step can be skipped. The output of the weighted averages becomes the AF's orientation in that frame of the animation. Continuity is not ensured, but the phrasing control technique of priority ramp at the beginning and end of each MU helps to minimize sudden changes in the rotation of any joint in the body during transitions of MUs.

The algorithms in the invention fill the need caused by the recently made available affordable PCs and game platforms. These platforms are now powerful enough to do exciting animations, including AF animation. However, they are not yet powerful enough to directly use the wealth of AF animation techniques that have been developed.

What the invention contributes is a user-based set of algorithms, that allow the end user of a game or educational software to interactively and intuitively choose and mix stored Motion Units (MU) in the course of running some application that moves articulated figures. Off-line, other well developed techniques can be used to create the MUs. These MUs are mixed and run on the low end platform using the fast algorithms discussed here.

Although illustrated and described herein with reference to certain specific embodiments, the present invention is nevertheless not intended to be limited to the details shown. Rather, various modifications may be made in the details within the scope and range of equivalents of the claims and without departing from the spirit of the invention.

Claims

1. A computer implemented method of manipulating a plurality of motion units to generate motion for an articulated figure in a sequence of frames in computer animation , said articulated figure including a plurality of joints and each motion unit defining the joint rotation angles for a particular motion , the method comprising the steps of ;

   assigning weights to each of the plurality of motion units wherein each of said plurality of joints in said articulated figure is made corresponding to the each weight assigned to each motion unit ,

   selecting a plurality of motion units ;

   combining the selected plurality of motion units in a manner that corresponding joint rotation angles of the articulated figures are mixed to each other by using said weights.

2. A method according to claim 1, wherein the step of combining includes combining the plurality of motion units in accordance with a calculation related to an average of the assigned weights.

3. A method according to claim 1, wherein the step of assigning weights includes assigning weights to the plurality of motion units according to a predetermined priority scheme.

4. A method according to claim 1, including the further step of creating and storing the plurality of motion units in a library prior to the step of selecting the plurality of motion units.

5. A method according to claim 1, including the further step of decreasing the motion of the articulated figure at a beginning and at an end of each of the plurality of motion units.

6. A method according to claim 1, including the further step of frame-to-frame smoothing which combines two or more frames using a weighted average to reduce discontinuities between adjacent frames.

7. A computer implemented system for manipulating a plurality of motion units to generate motion for an articulated figure for frames in computer animation, said articulated figure including a plurality of joints and each motion unit defining the joint rotations for a particular joint, the system comprising:
selection means for selecting a plurality of motion units;
means for assigning weights to each of the plurality of motion units wherein each of said plurality of joints in said articulated figure corresponds to an individual weight assigned to each motion unit; and
means for combining each of the selected plurality of motion units, based on the individual weights, to define the position of the corresponding joints of the articulated figure to generate a motion in the articulated figure corresponding to the selected motion units.

8. A system according to claim 7, wherein the means for combining each of the plurality of motion units includes further means for calculating an average of the assigned weights for each of the plurality of motion units.

9. A system according to claim 7, wherein the means for assigning weights includes further means for prioritizing the plurality of motion units according to a predetermined priority.

10. A system according to claim 7, further comprising means for creating and storing the plurality of motion units in a library.

11. A system according to claim 7, further comprising means for decreasing the motion of the articulated figure at a beginning and at an end of each of the plurality of motion units.

12. A system according to claim 7, further comprising frame-to frame smoothing means to smooth said motion which combines two or more frames using a weighted average to reduce discontinuities between adjacent frames.

13. A method according to claim 1, wherein the articulated figure represents a human and one of said plurality of motion units indicates a degree of fatigue of the articulated figure.

14. A method according to claim 1, wherein one of said plurality of motion units is an idle motion unit that controls the motion of the articulated figure only when the user has not selected any other motion unit.

15. A method according to claim 1, wherein each of the selected motion units are added to a queue of concurrently running motion units.

16. A method according to claim 1, further comprising:
assigning frame weights to each frame in said sequence of frames;
wherein the calculation used to combine the plurality of motion units is based on the weights assigned to each motion unit and the frame weights.

17. A system according to claim 7, wherein the articulated figure represents a human and one of said plurality of motion units indicates a degree of fatigue of the articulated figure.

18. A system according to claim 7, wherein each one of said plurality of motion units is an idle motion unit that controls the motion of the articulated figure only when the user has not selected any other.

19. A system according to claim 7, wherein each of the selected motion units are added to a queue of concurrently running motion units.

20. A system according to claim 7, further comprising:
means for assigning frame weights to each frame in said sequence of frames;
wherein the calculation performed by the means for combining the plurality of motion units is based on the weights assigned to each motion unit and the frame weights.

FIG. 1

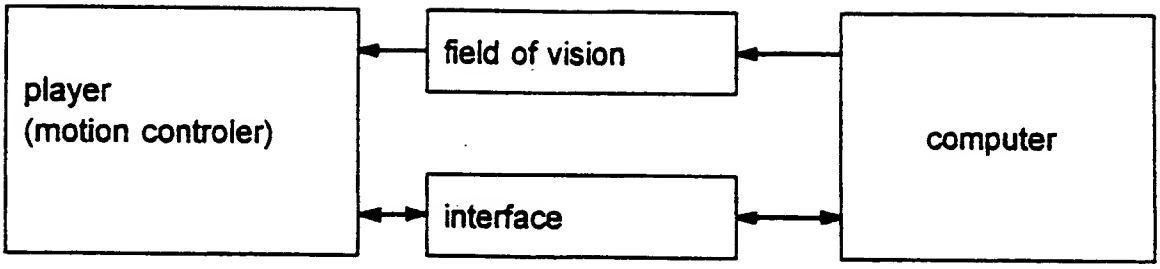

player
(motion controler)
field of vision
interface
computer

FIG. 2

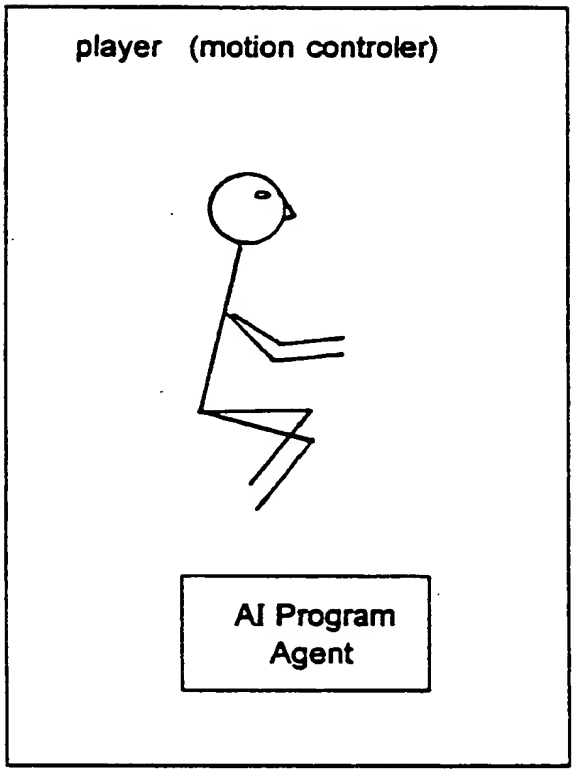


FIG. 3 (a)

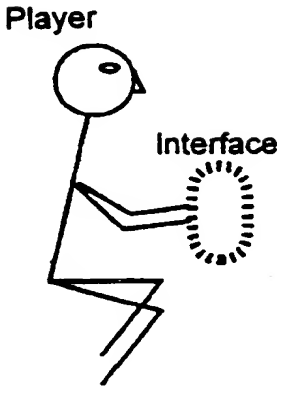


FIG. 3(b)

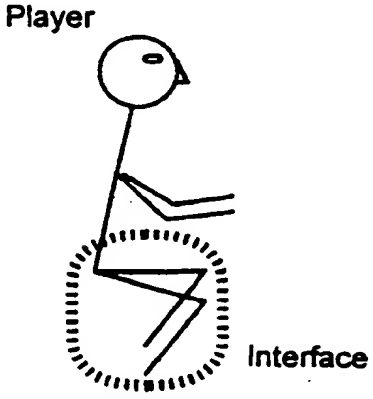


FIG. 3(c)

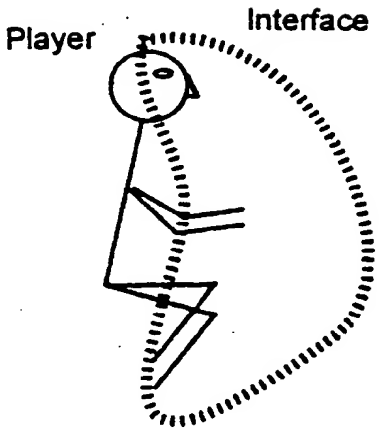


FIG. 4 (a)

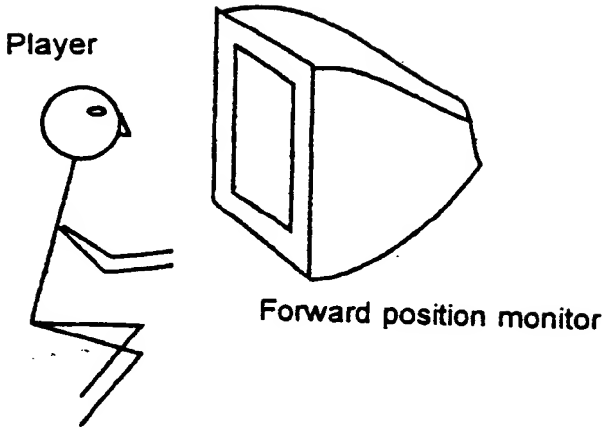


FIG. 4 (b)

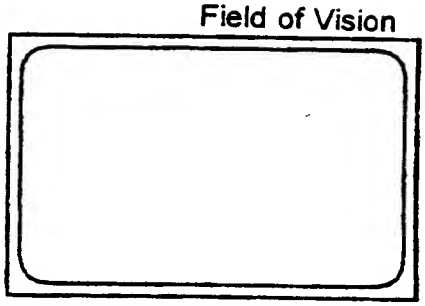


FIG. 5 (a)

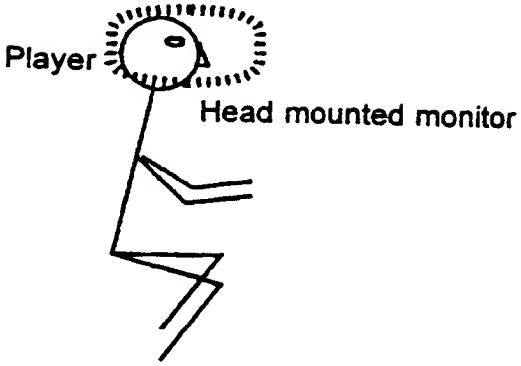


FIG. 5 (b)

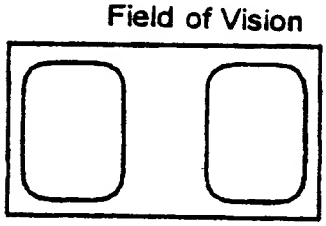


FIG. 6(a)

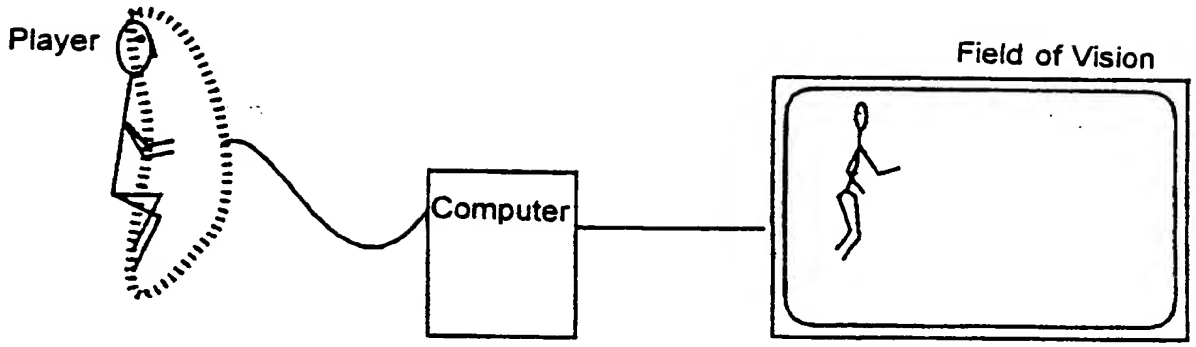

Player
Field of Vision
Computer

FIG. 6(b)

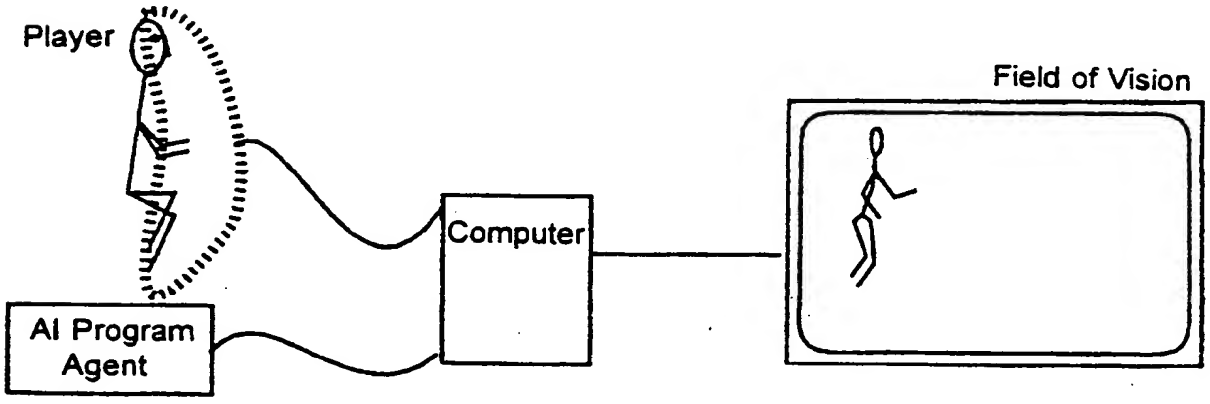

Player
Field of Vision
Computer
AI Program
Agent

FIG. 7(a)

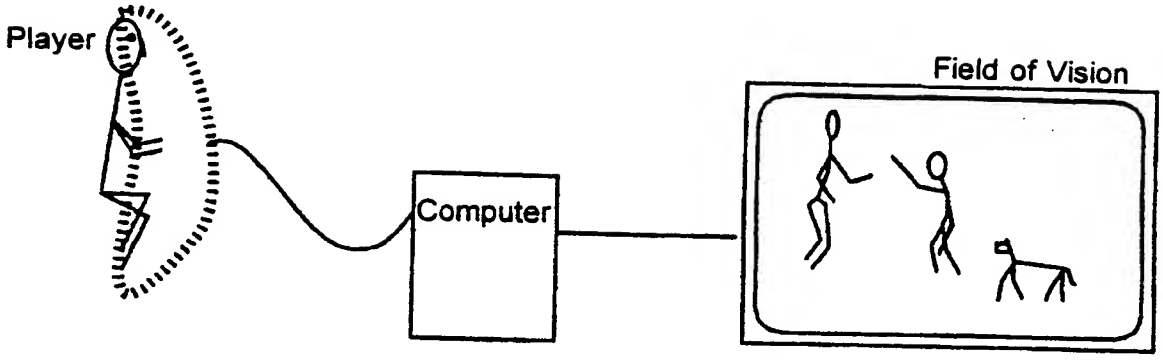

Player
Computer
Field of Vision

FIG. 7(b)

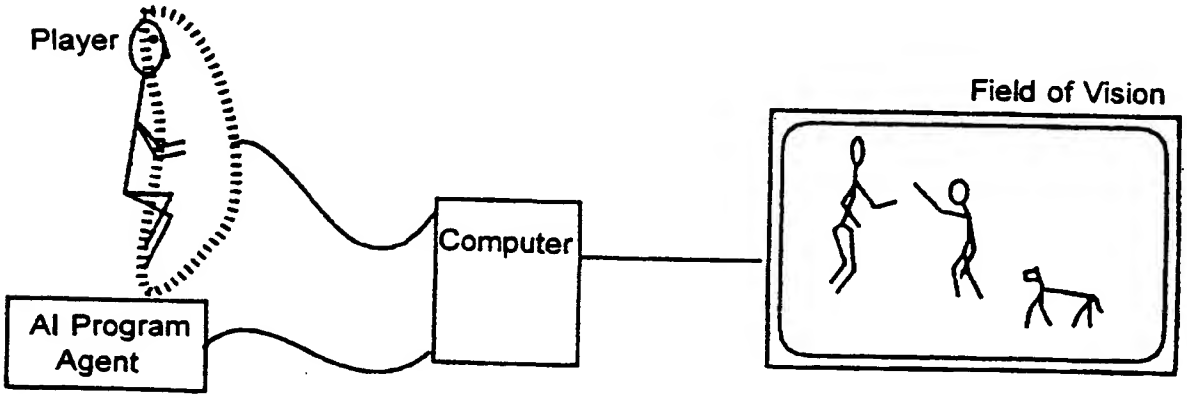

Player
AI Program
Agent
Computer
Field of Vision

FIG. 8(a)

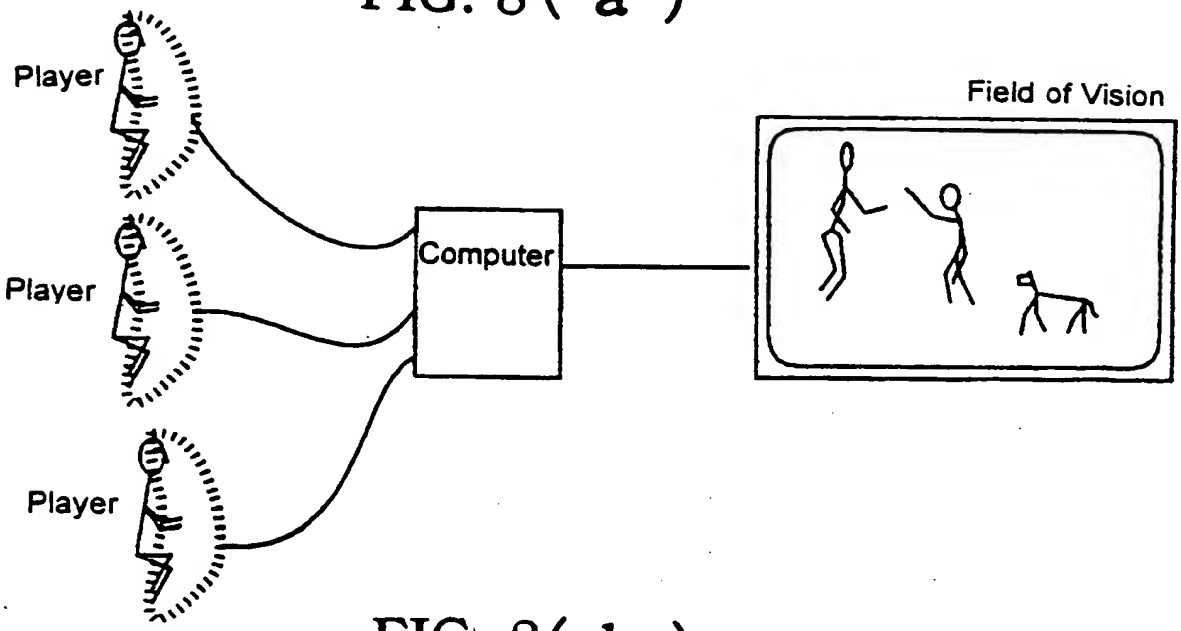


FIG. 8(b)

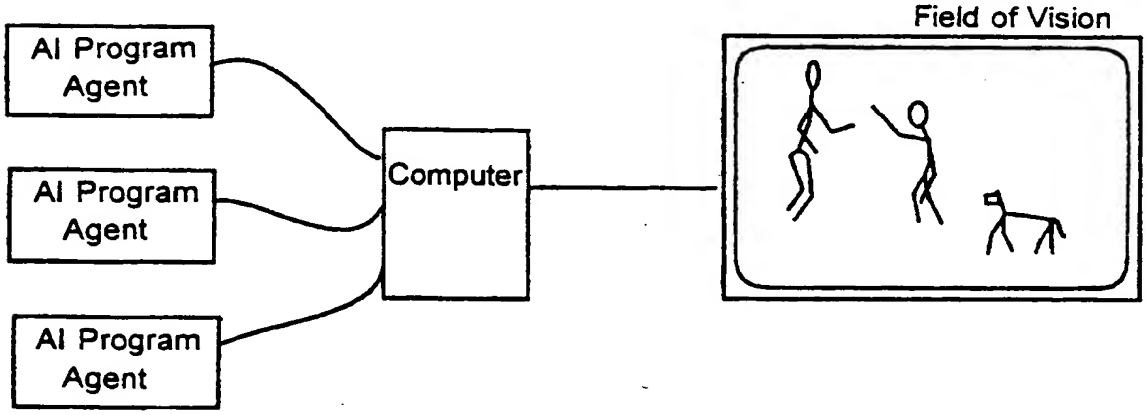


FIG. 8(c)

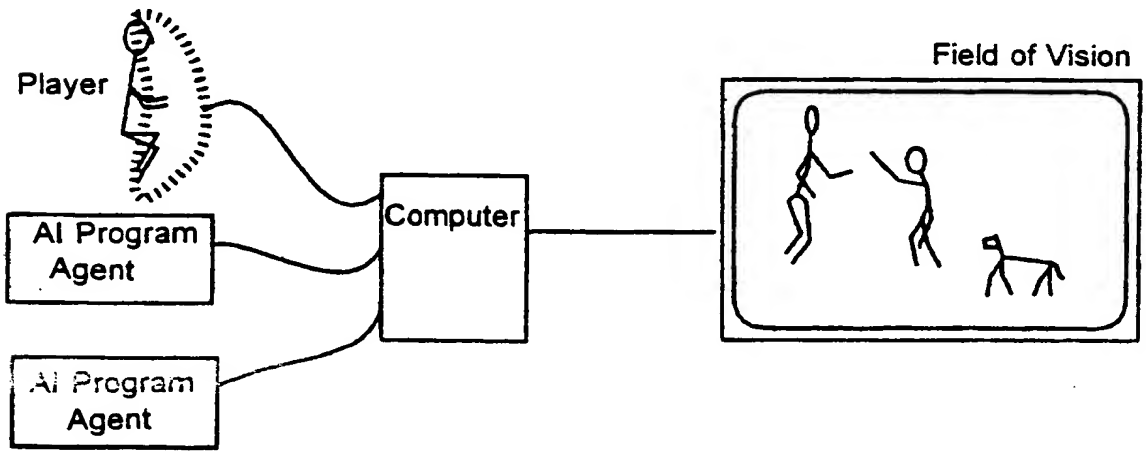


FIG. 9(a)

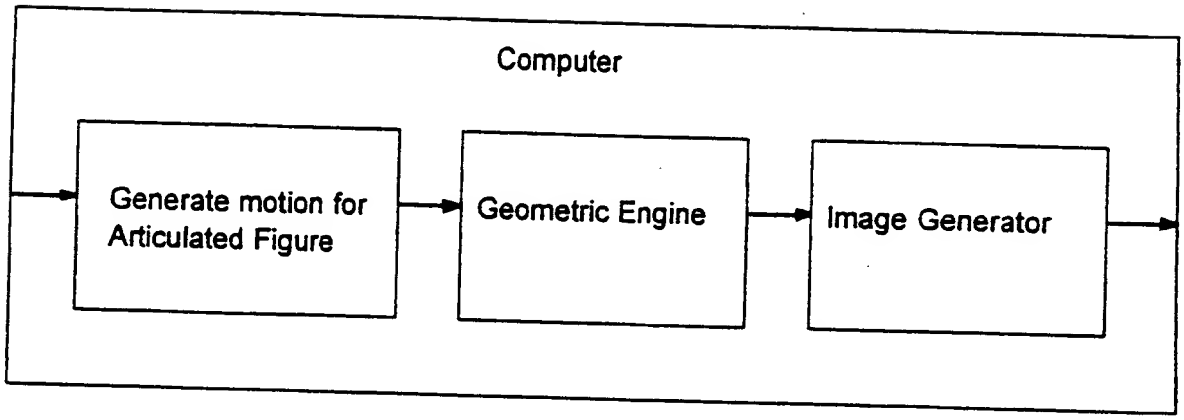

Computer
Generate motion for Articulated Figure
Geometric Engine
Image Generator

FIG. 9 (b)

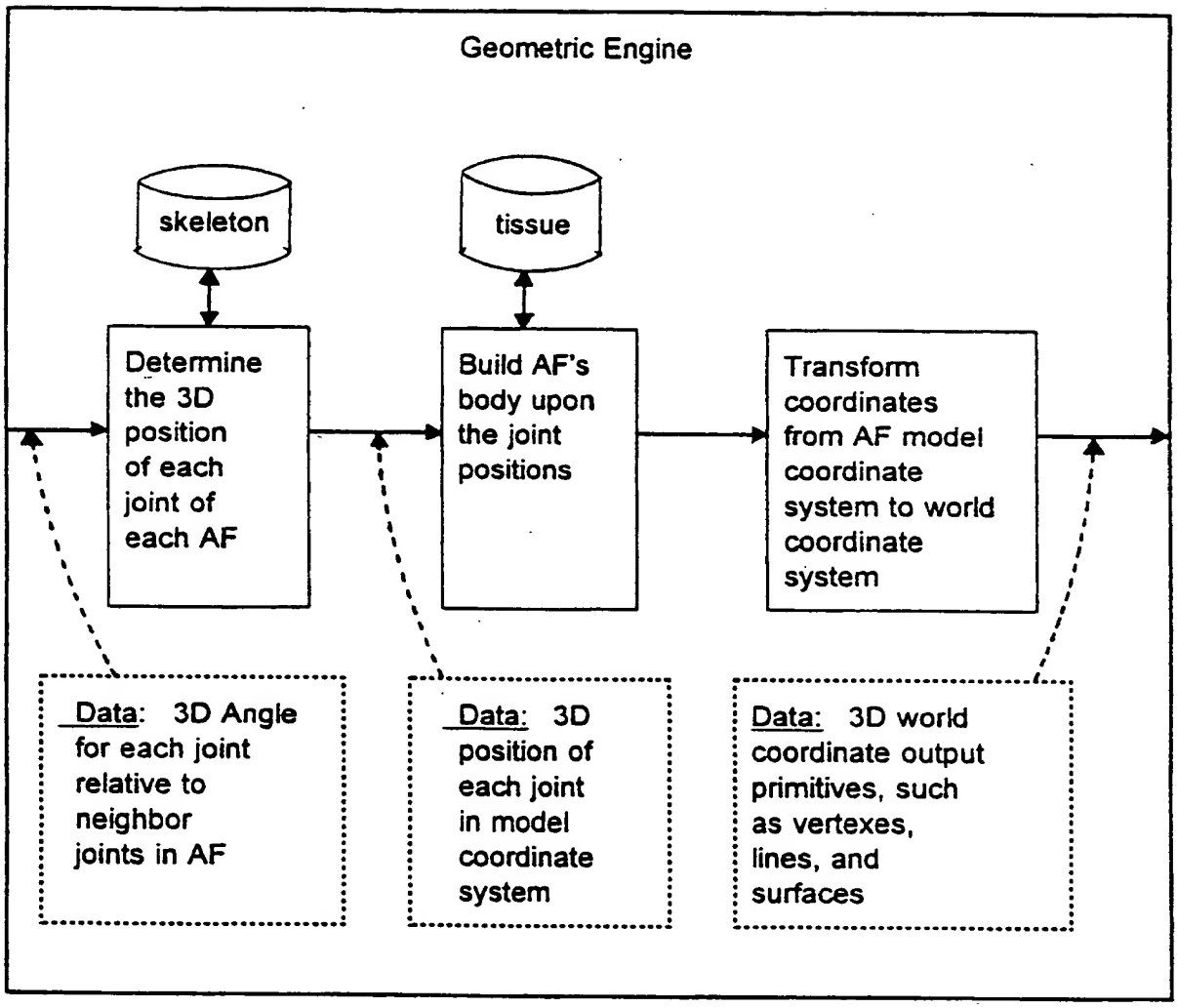

Geometric Engine
skeleton
tissue
Determine the 3D position of each joint of each AF
Build AF's body upon the joint positions
Transform coordinates from AF model coordinate system to world coordinate system
Data: 3D Angle for each joint relative to neighbor joints in AF
Data: 3D position of each joint in model coordinate system
Data: 3D world coordinate output primitives, such as vertexes, lines, and surfaces

FIG. 9 (c)

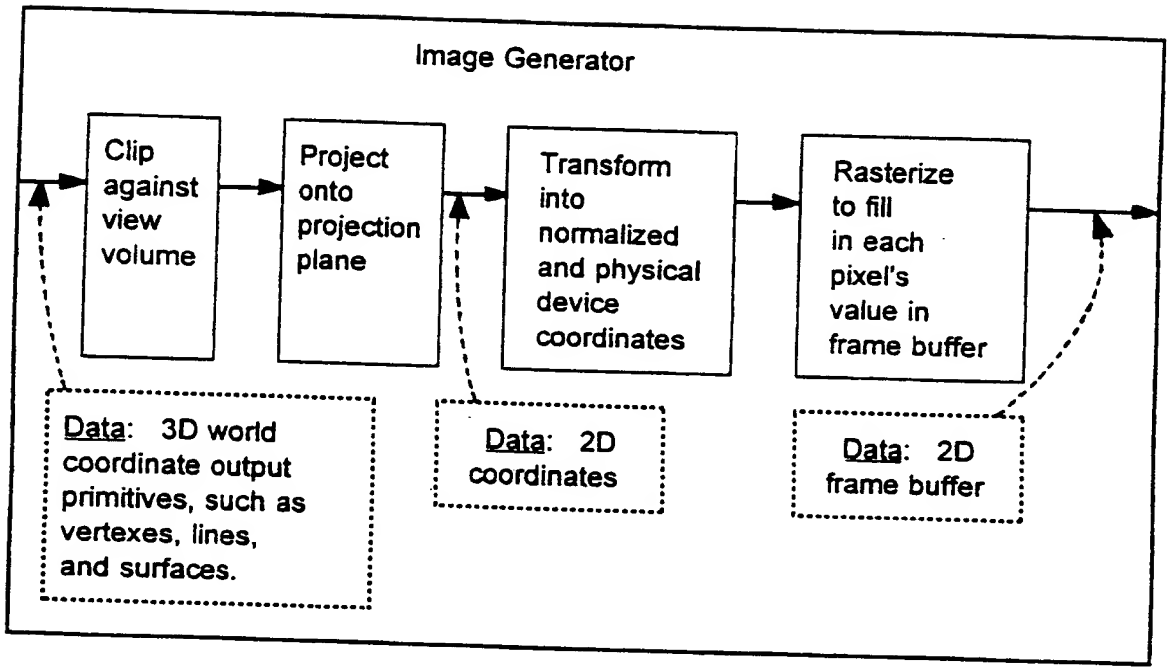


Image Generator
Clip against view volume
Project onto projection plane
Transform into normalized and physical device coordinates
Rasterize to fill in each pixel's value in frame buffer
Data: 3D world coordinate output primitives, such as vertexes, lines, and surfaces.
Data: 2D coordinates
Data: 2D frame buffer

Joints: Articulated Figures

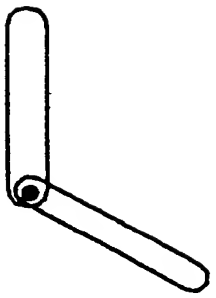


FIG. 10 (a)

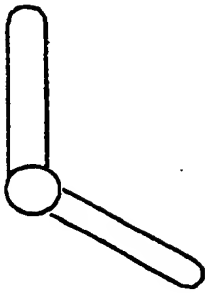


FIG. 10 (b)

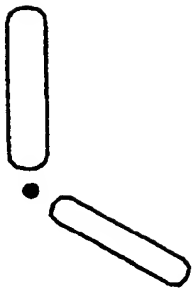


FIG. 10 (c)

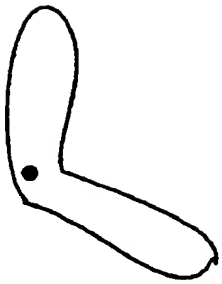


FIG. 10 (d)

FIG. 11(a)

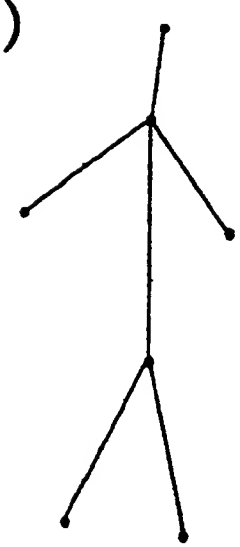


FIG. 11(b)

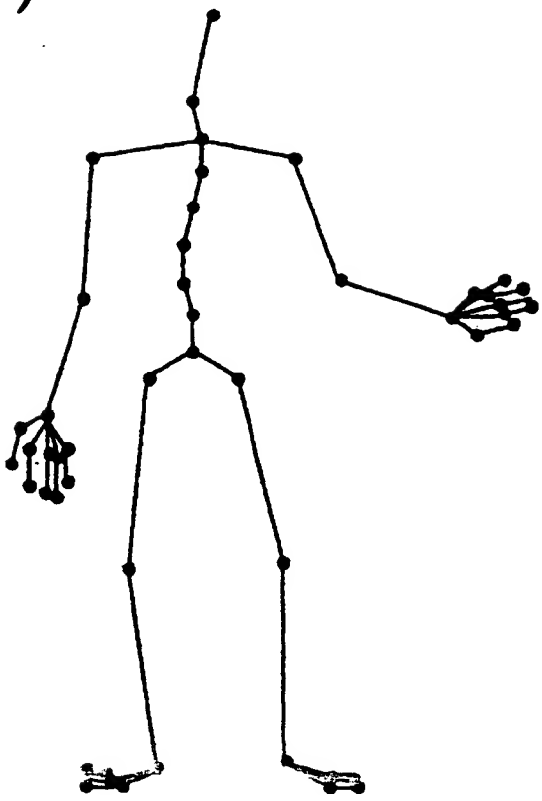


FIG. 12 (a)

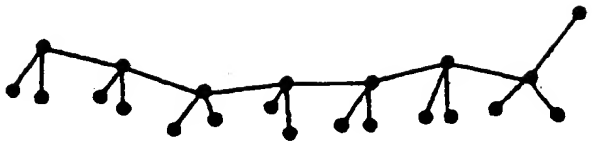


FIG. 12 (b)

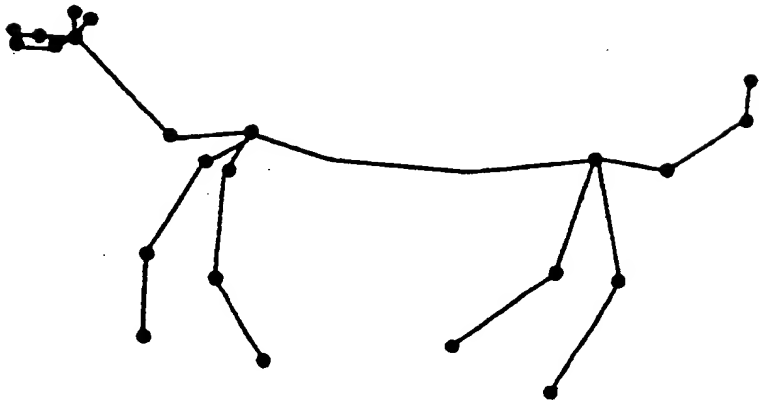


FIG. 13

Example Where AF is Covered With a Surface

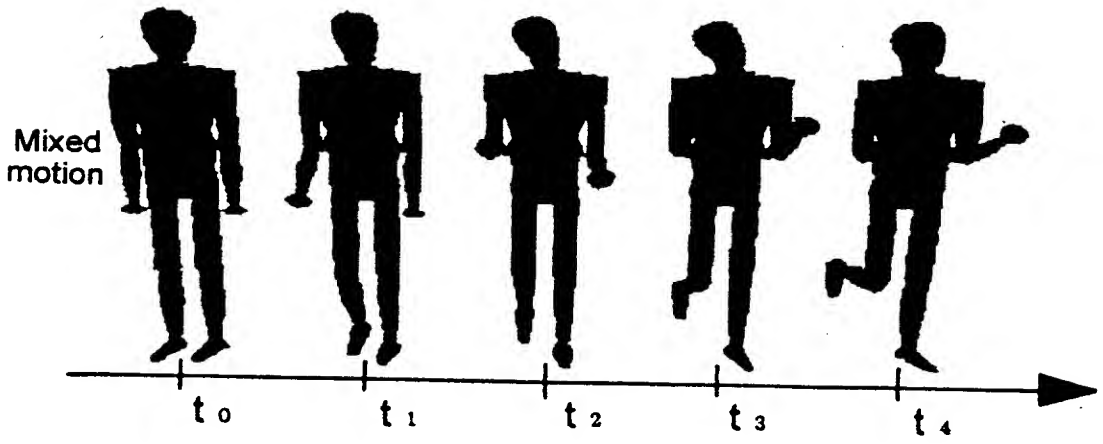


FIG. 14 (a)

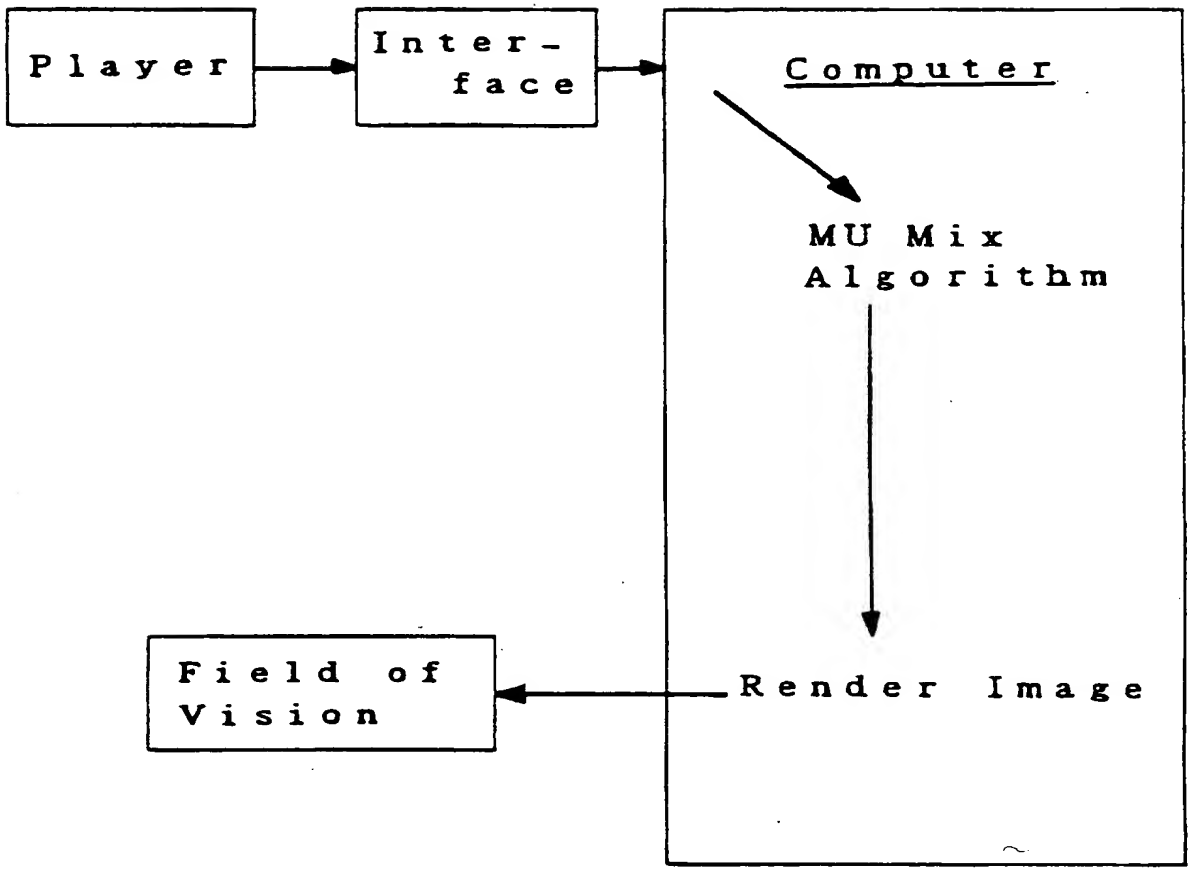

Player
Inter-face
Computer
MU Mix Algorithm
Render Image
Field of Vision

FIG. 14(b)

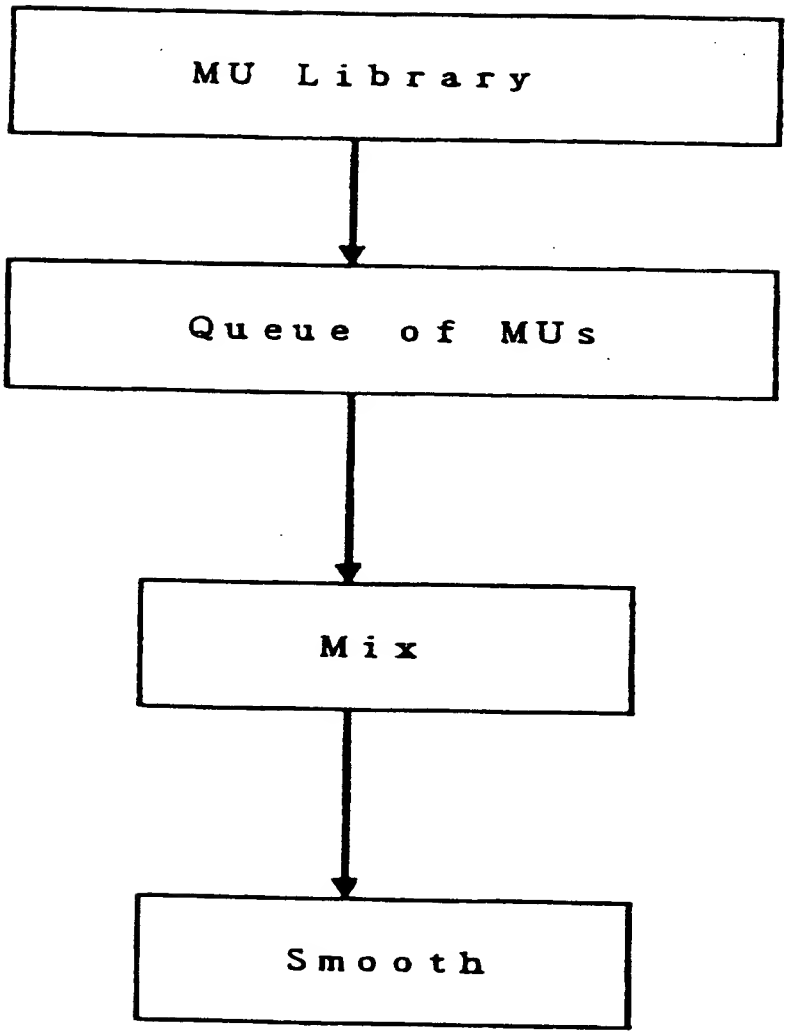


FIG. 14(C)

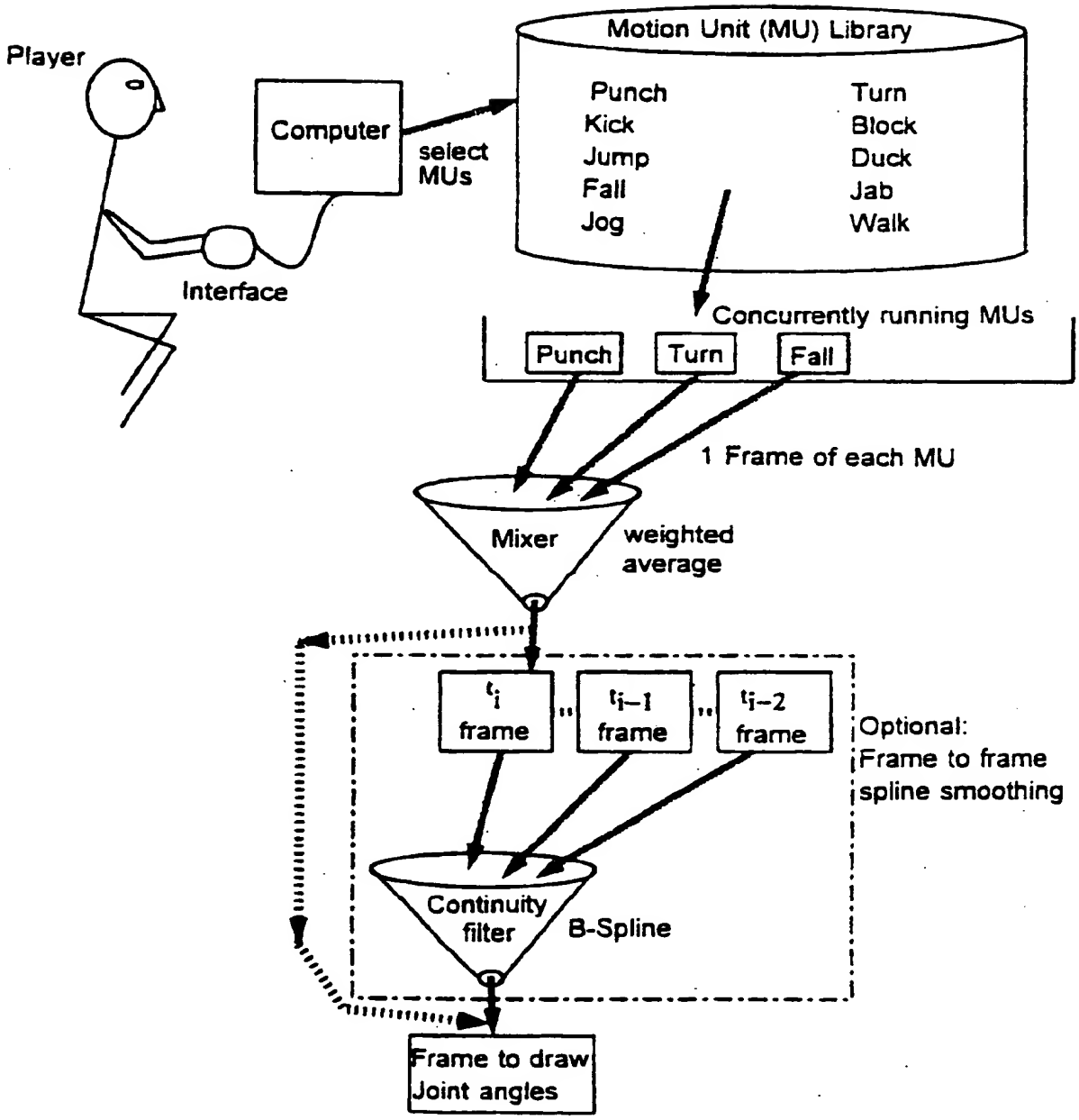

Player
Computer
select MUs
Interface
Motion Unit (MU) Library
Punch
Kick
Jump
Fall
Jog
Turn
Block
Duck
Jab
Walk
Concurrently running MUs
Punch
Turn
Fall
1 Frame of each MU
Mixer
weighted average
t_i frame
t_{i-1} frame
t_{i-2} frame
Optional: Frame to frame spline smoothing
Continuity filter
B-Spline
Frame to draw Joint angles

FIG. 15

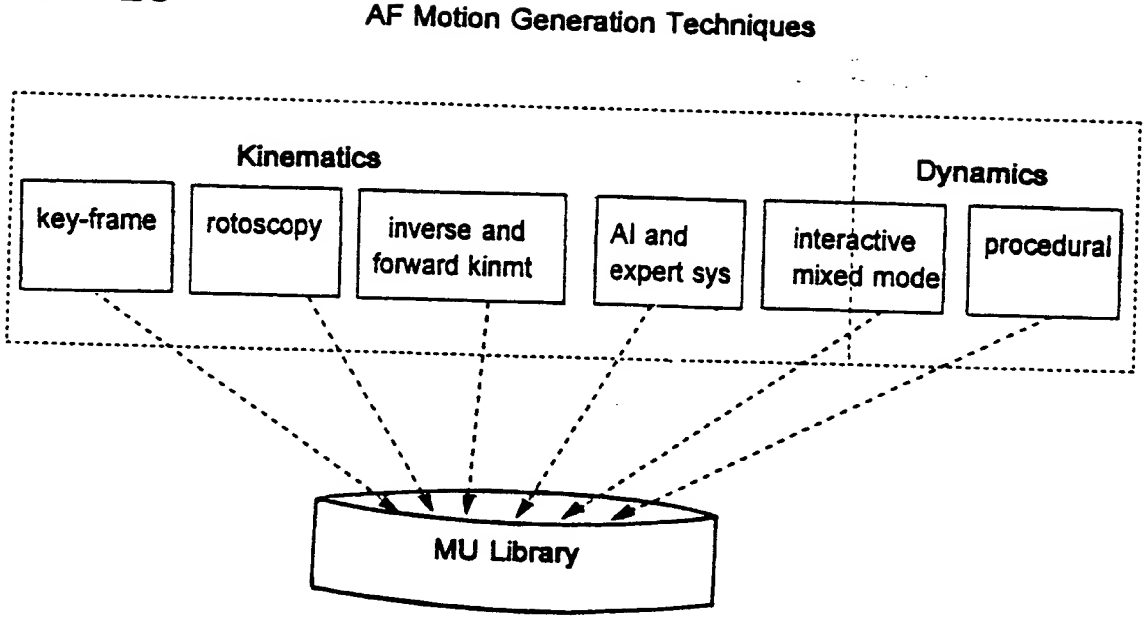
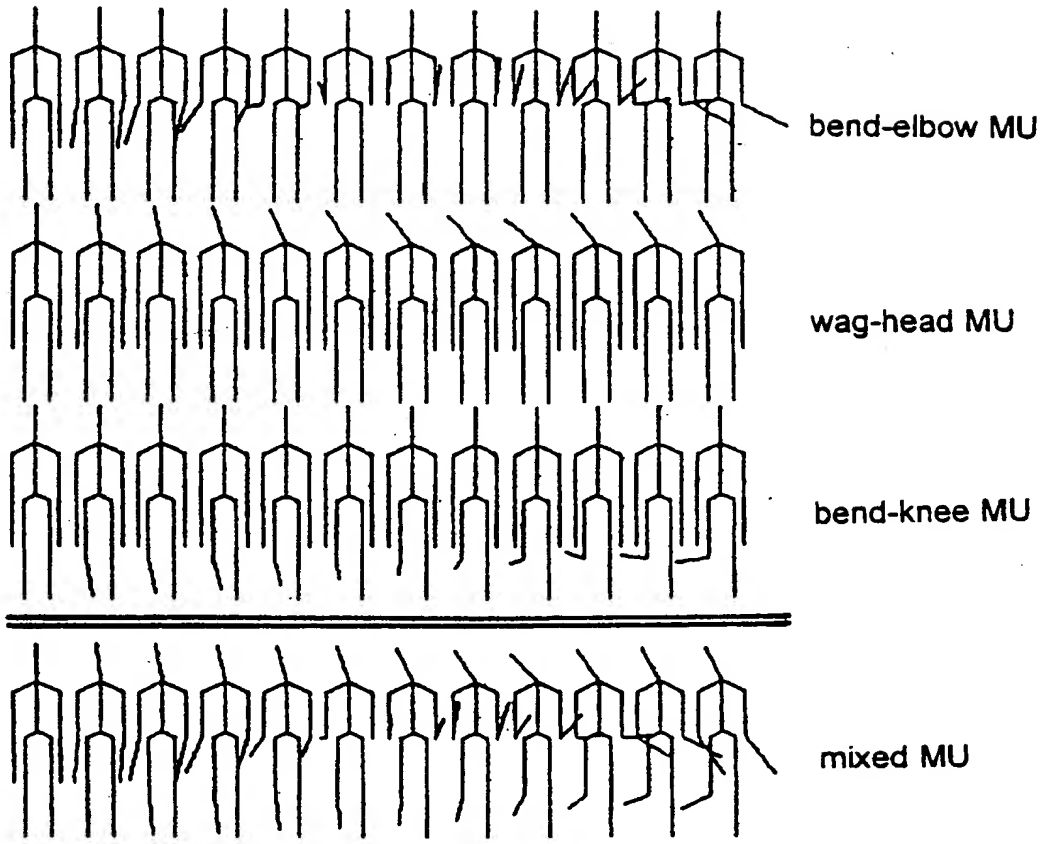

AF Motion Generation Techniques
Kinematics
Dynamics
key-frame
rotoscopy
inverse and forward kinmt
AI and expert sys
interactive mixed mode
procedural
MU Library

FIG. 16(a)

Example of Mixed MUs



Frames at: t_0 t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10} t_{11}

FIG. 16 (b)

$$\theta_{iPJ} = \frac{(W_{iP})(W_{iJ})(\theta_{iPJ})}{(W_{iP})(W_{iJ})} + \frac{(W_{iP})(W_{iJ})(\theta_{iPJ})}{(W_{iP})(W_{iJ})} + \frac{(W_{iP})(W_{iJ})(\theta_{iPJ})}{(W_{iP})(W_{iJ})}$$

one joint angle of mixed MU

from bend-elbow MU

from wag-head MU

from bend-knee MU

FIG. 16(c)

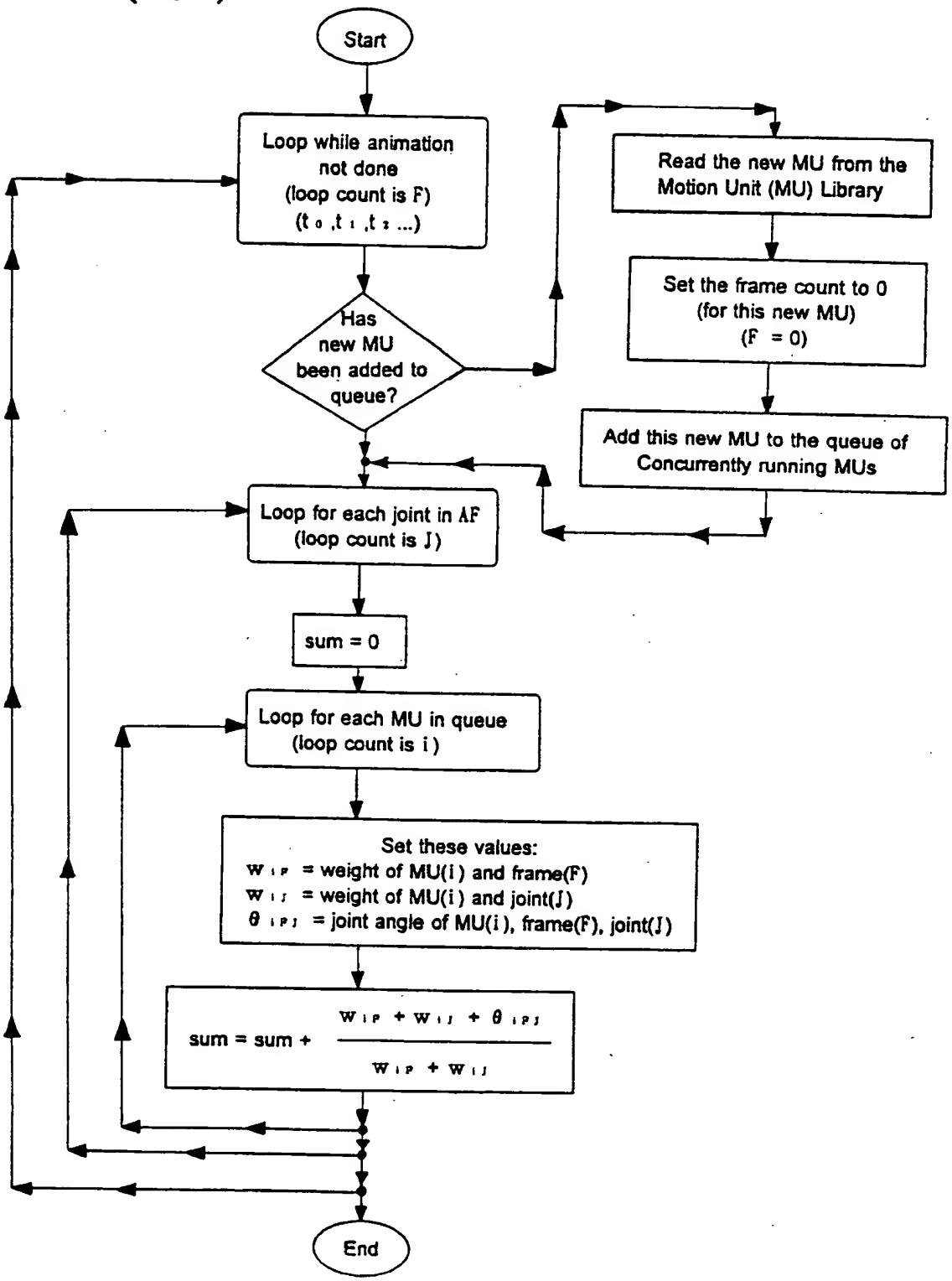

Start
Loop while animation not done (loop count is F) (t0 ,t1 ,t2 ...)
Has new MU been added to queue?
Read the new MU from the Motion Unit (MU) Library
Set the frame count to 0 (for this new MU) (F = 0)
Add this new MU to the queue of Concurrently running MUs
Loop for each joint in AF (loop count is J)
sum = 0
Loop for each MU in queue (loop count is i)
Set these values:
wiF = weight of MU(i) and frame(F)
wiJ = weight of MU(i) and joint(J)
θiFJ = joint angle of MU(i), frame(F), joint(J)
sum = sum + (wiF + wiJ + θiFJ) / (wiF + wiJ)
End

FIG. 17

MU File Specification: Joint Rotations
– a series of frame data
– each frame consists of a long vector of rotation angles for each joint: (Θ_z, Θ_x, Θ_y)

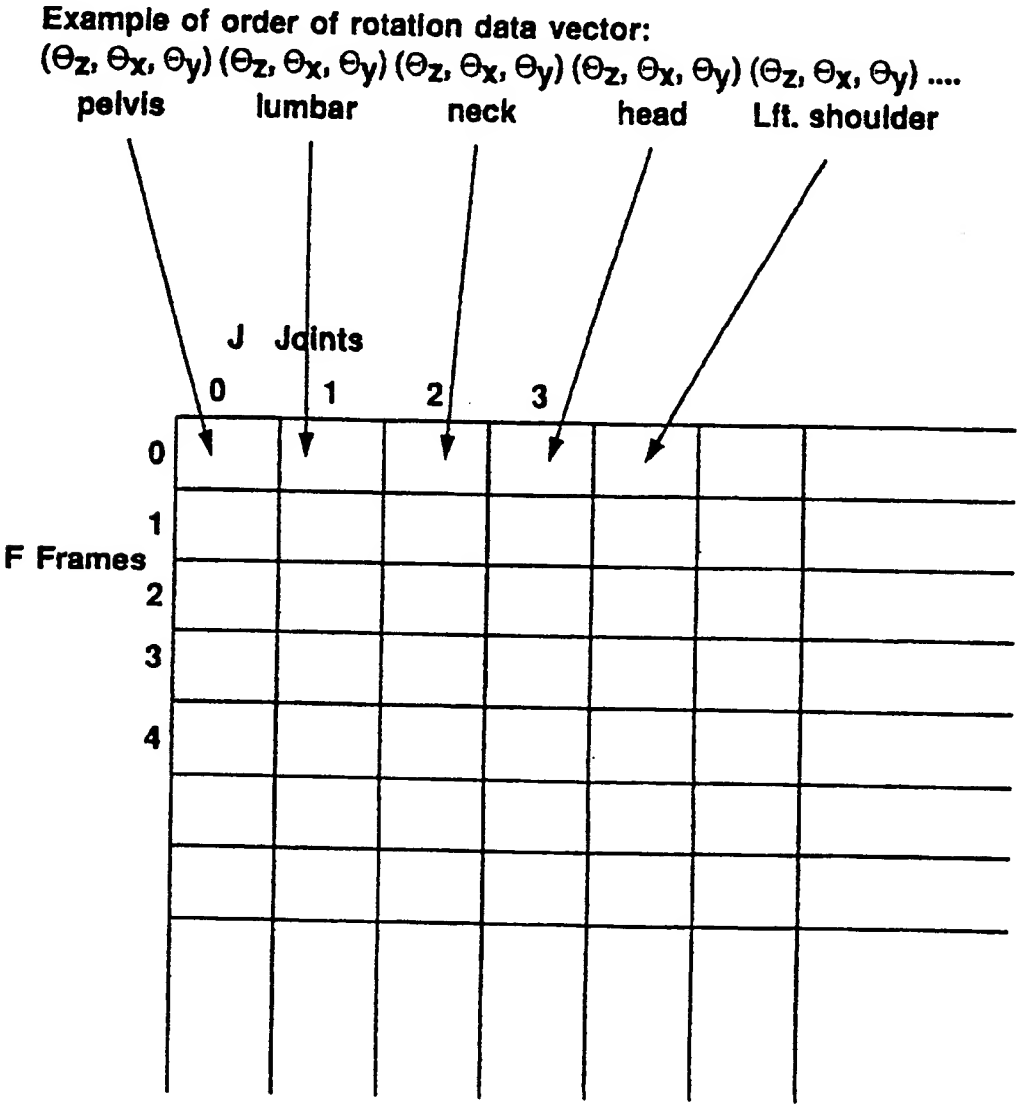


FIG. 18

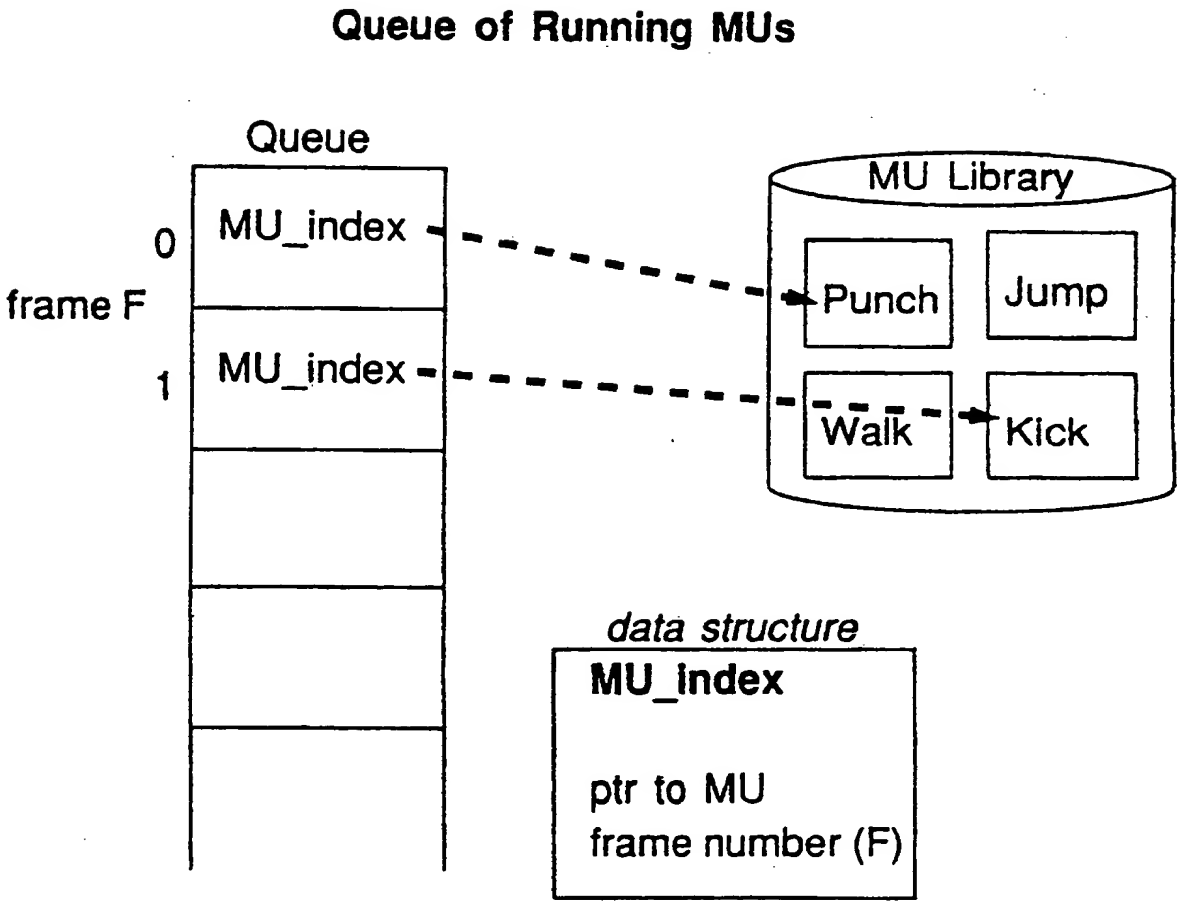

Queue of Running MUs
Queue
MU_index
0
frame F
MU_index
1
MU Library
Punch
Jump
Walk
Kick
data structure
MU_index
ptr to MU
frame number (F)

FIG. 19

Examples of Frame Weight Patterns for a MU

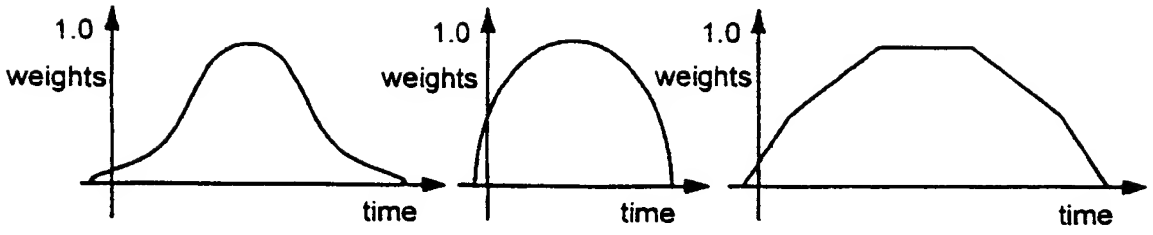


FIG. 20

Example of Frame Weight Pattern for a MU

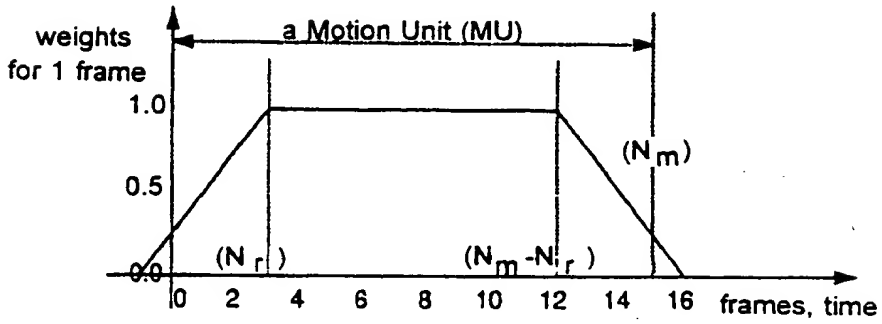


FIG. 21

Phrase Control Example

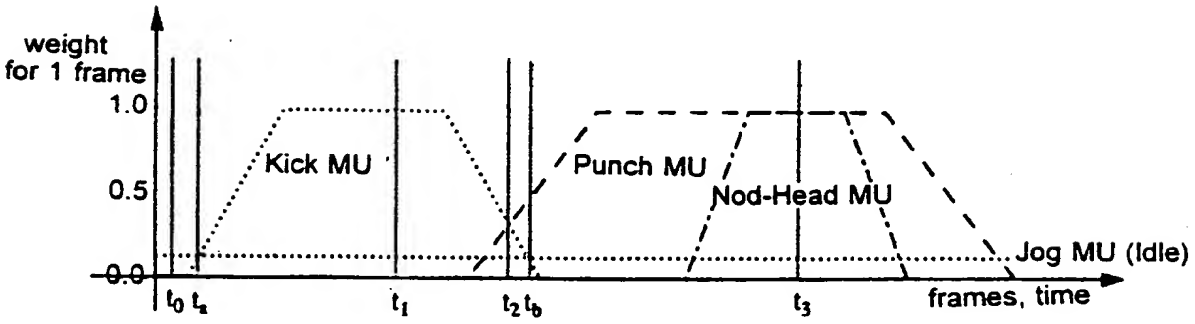
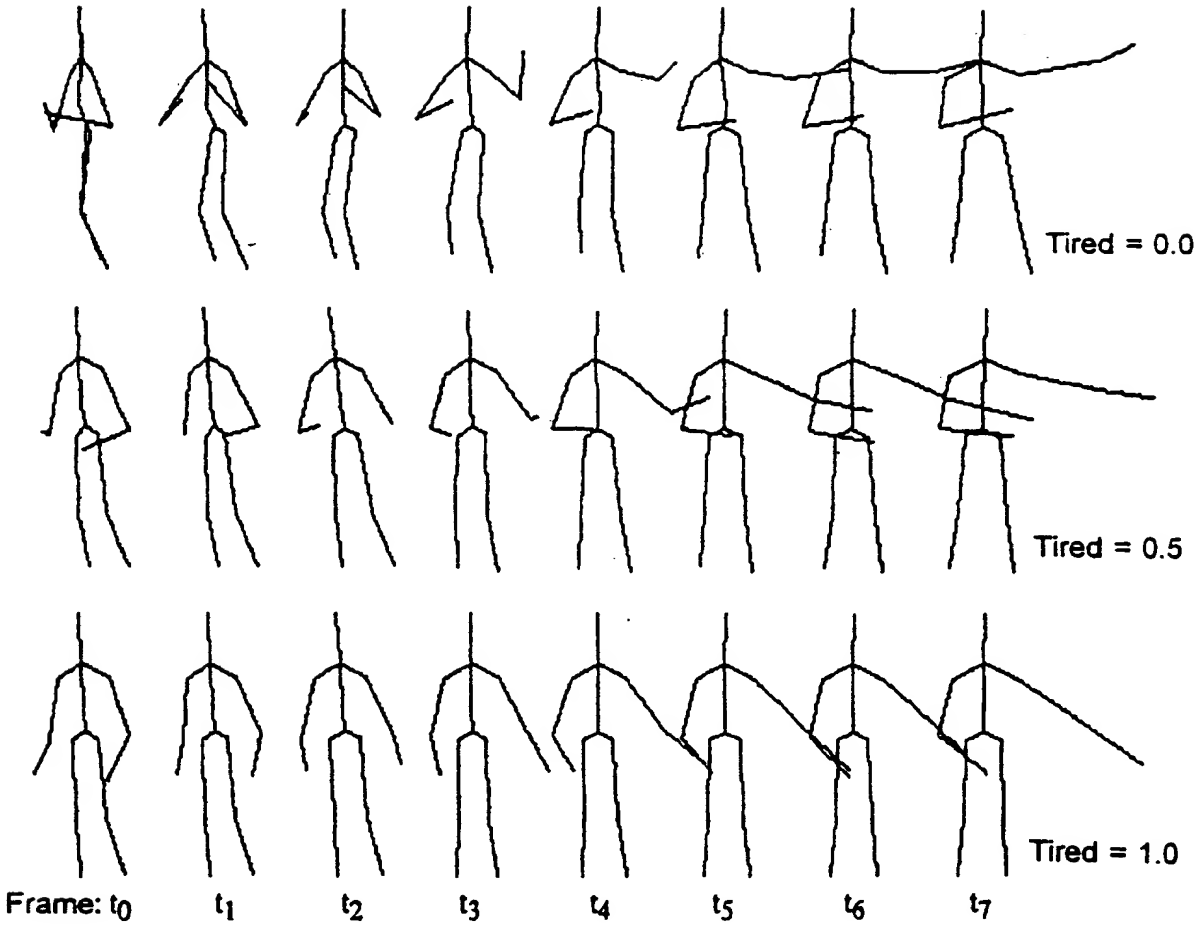


FIG. 22



(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 712 097 A3

(12) EUROPEAN PATENT APPLICATION

(88) Date of publication A3:
05.06.1996 Bulletin 1996/23

(43) Date of publication A2:
15.05.1996 Bulletin 1996/20

(21) Application number: **95117529.8**

(22) Date of filing: **07.11.1995**

(51) Int. Cl.[6]: **G06T 15/70**

(84) Designated Contracting States:
DE FR GB

(30) Priority: **10.11.1994 US 337566**

(71) Applicant: **MATSUSHITA ELECTRIC INDUSTRIAL CO., LTD.**
Kadoma-shi, Osaka 571 (JP)

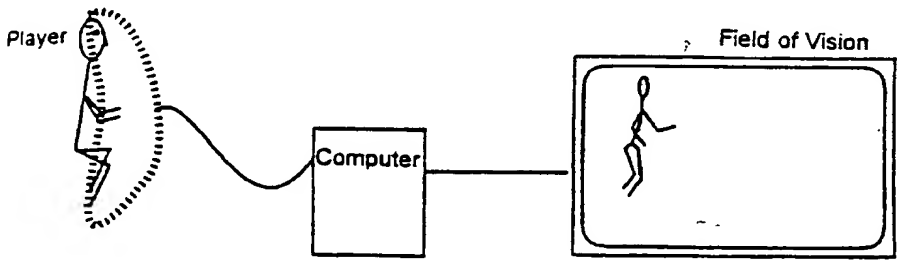
(72) Inventors:
- **Dow, Douglas Eric**
  2368 Singapore (SG)
- **Inada, Kazuhiko**
  Kadoma-shi, Osaka 571 (JP)

(74) Representative: **Grünecker, Kinkeldey, Stockmair & Schwanhäusser Anwaltssozietät**
Maximilianstrasse 58
80538 München (DE)

(54) Method and system for manipulating motion units for computer articulated figure animation

(57) A method and system for manipulating and mixing previously stored motion units for computer articulated-figure animation. The motion units are combined in a weighted average. The weights give a priority to certain joint rotations of a motion unit which constitute the essence of that motion unit. The motion units can be mixed in an asynchronous fashion, with no limit on the number of motion units that can be mixed together. Phrasing control assigns a weight to each frame in a motion unit, and helps to ease the transitions at the beginning and end of a motion unit. Frame-to-frame smoothing is also used. Fatigue of an articulated-figure can be simulated by the addition of a "tired" motion unit.

FIG. 6(a)



EP 0 712 097 A3

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 95 11 7529

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
A	SYSTEMS & COMPUTERS IN JAPAN, vol. 21, no. 7, 1990 NEW YORK US, pages 63-74, XP 000172928 KIMOTO AND YASUDA 'A METHOD FOR FRAME REPRESENTATION OF MOVING OBJECTS FOR KNOWLEDGE-BASED CODING' * the whole document *	1,7	G06T15/70

TECHNICAL FIELDS SEARCHED (Int.Cl.6)

G06T

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	3 April 1996	Burgaud, C

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)